UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

SEGURIDAD CENTRADA EN LA INFORMACIÓN PARA VEHÍCULOS AÉREOS NO TRIPULADOS

PROYECTO FIN DE GRADO

Autor: Carlos Mariano Castillo Mateos

Tutor: Iván Vidal Fernández

22 de junio de 2016

Agradecimientos

Me gustaría dar las gracias a mi tutor Iván, por su infinita paciencia, su exigencia y por saber iluminarme cuando pensaba que todo era oscuro.

A todos los compañeros que me han acompañado en los diferentes cursos del grado, y a los nuevos amigos que me llevo.

A todos los profesores que se han interesado por mi desarrollo tanto educativo como personal.

A mi familia, que sin su incondicional apoyo no habría podido llegar hasta aquí.

A mis amigos, Joaquín, Pablo, Lucía, Javi, Fazio, Laura e infinitos nombres más, por apoyarnos mutuamente en esta locura.

Y sobre todo a Carlota, sin ti este proyecto no habría sido posible. Gracias por estar siempre apoyándome, y ayudarme a superar cada obstáculo que se me ha presentado.

A todos, muchísimas gracias.

Resumen

Actualmente, con la evolución de la tecnología podemos ver más frecuentemente el uso de Unmanned Aerial Vehicles (UAVs) con diferentes propósitos. Los medios de comunicación informan cómo los cuerpos militares de diferentes países usan estos dispositivos en operaciones donde la vida de un soldado puede correr riesgo o para acceder a lugares inaccesibles para las personas. También se están haciendo más frecuentes las noticias sobre grandes compañías tecnológicas que están investigando sobre el uso de estos dispositivos en el ámbito civil, en el transporte de paquetes, reduciendo el coste y el tiempo de desplazamiento; asistencia en operaciones de emergencia, como incendios; operaciones de guardacostas, como en la monitorización de vías marítimas; conservación del medio ambiente; y muchas otras operaciones. Aunque, hoy en día dónde más se utilizan los UAVs, es como objeto de diversión orientado a personas de todas las edades.

¿Pero cómo podemos garantizar la seguridad en la entrega de información en entornos de UAVs? Las soluciones actuales para proporcionar seguridad en red permiten proteger conexiones entre equipos, o canales de comunicación. Sin embargo, con este esquema, un consumidor interesado en recibir un contenido de forma segura no puede garantizar la autenticidad del mismo a no ser que lo obtenga directamente de su fuente original.

En este trabajo se aborda esta temática, explorando el uso de los nuevos conceptos de seguridad basada en contenido para garantizar la distribución segura de información en entornos de UAVs. En este documento se presenta el estado del arte, el diseño y la implementación del proyecto y las pruebas realizadas.

Abstract

Nowadays, with the evolution of technology we can see more frequently the use of UAVs for multidisciplinary purposes. We can see how the military of different countries are using them in many operations where the life of a soldier is at risk or to access places that are impossible to people. Also we are hearing news about researches of technology companies in civil matters using the UAVs to transport packets, reducing the cost and time of displacement; assistance in emergency operations, as fires; coastguards operations, as sea-line monitoring: conservation of the environment; and so many others. But where we can see a higher demand on UAVs are in the usage for joy purpose, with people in all ages.

But, how can we guarantee the safety delivery of the information in UAVs environment?. The current solutions of network security allows secured connections between computer equipments, or in communication channels. Nevertheless, with this design, an interested consumer in receiving the data secured, cannot guarantee the authenticity of this data unless it is received from the original source.

In this project this topic is discussed, exploring the use of the new security concepts based on the content to guarantee the secure distribution of information in UAVs environment. In this document it is discussed the state of art, the design and implementation of the project, and performed test.

Información Complementaria

Según la indicación de la normativa:

"Los alumnos del plan 2011 deben demostrar la competencia del idioma inglés en su TFG. Los alumnos que estén cursando el grados del Plan 2011 deberán redactar en inglés de forma obligatoria en la memoria del TFG un resumen extendido (5-10 páginas), los capítulos de introducción y conclusiones, (siendo recomendable la escritura de la memoria completa en inglés) para cumplir con los objetivos de aprendizaje marcados en la ficha de la asignatura TFG."

Debido a esto, los capítulos de *Introducción* y *Conclusión* así como el resumen extendido de la memoria, pueden ser encontrados en forma de apéndice al final del documento.

Listado de acrónimos

ACK Acknowledgement.

AES Advanced Encryption Standard.

AESA Agencia Estatal de Seguridad Aérea.

AFCS Automatic Flight Control System.

API Application Programming Interface.

CCN Content-Centric Networking.

CDF Cumulative Distribution Function.

CDN Content Distribution Network.

DES Data Encryption Standard.

DNS Domain Name System.

DS Distribution System.

FDM Frequency Division Multiplexing.

GCS Ground Control Station.

ICN Information-Centric Networks.

IDEA International Data Encryption Algorithm.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IPSec Internet Protocol Security.

JDK Java Development Key.

JRE Java Runtime Environment.

KDC Key Distribution Center.

MAC Message Authentication Code.

MD5 Message-Digest Algorithm 5.

MDC Modification Detection Codes.

OSI Open System Interconnection.

P2P Peer-to-Peer.

RAM Random Access Memory.

RPA Remotely Piloted Aircraft System.

RSA Rivest, Shamir and Adleman.

RTCP Real-time Transport Control Protocol.

RTP Real-time Transport Protocol.

SDP Session Description Protocol.

SHA Secure Hash Algorithm.

SIP Session Initiation Protocol.

TCP Transmission Control Protocol.

TDM Time Division Multiplexing.

UAS Unmanned Aircraft Systems.

UAV Unmanned Aerial Vehicle.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

VANT Vehículo Aéreo No Tripulado.

VoIP Voice Over Internet Protocol.

XOR Exclusive OR.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se discuten los principales aspectos del proyecto, así como los objetivos, el alcance y la motivación para llevarlo a cabo. Posteriormente, se introduce la estructura del documento.

1.1. Motivación del Proyecto

Un drone o UAV es un Vehículo Aéreo No Tripulado (VANT), como su nombre indica, es un vehículo aéreo que no necesita una tripulación a bordo, pero sí necesita un operario en una base de control terrestre o Ground Control Station (GCS).

Existen diversos tipos de UAVs dependiendo del objetivo para los que son diseñados, con diferentes alcances de cobertura de acción como distintas capacidades dependiendo de su carga de pago (cámaras infrarrojas de vídeo, sensores térmicos, radares, sensores de frecuencias, y demás aparatos electrónicos). Estas capacidades permiten ejecutar múltiples actividades, tradicionalmente de índole militar, como seguimiento de flotas enemigas, monitorización del objetivo, localización y destrucción de minas terrestres, interferencia y destrucción de sistemas de radares.

En el ámbito civil, el desarrollo de UAVs ha supuesto una revolución, ya que gracias a la evolución de la tecnología, el uso de UAV permite la creación de mapas en 3D, monitorización en procesos de construcción, producción de películas; haciendo que cada vez un público más diverso sepa de su existencia y utilidad. En la figura 1.1 se puede apreciar el crecimiento esperado de UAVs sólo en los Estados Unidos de América.

En muchas de las aplicaciones descritas anteriormente es necesario garantizar la entrega, no sólo eficiente sino también segura de información. Sin embargo, las soluciones existentes para proporcionar seguridad en red presentan limitaciones, como proveer solamente seguridad a nivel de canal en la comunicación entre equipos y no a nivel de datos transferidos.

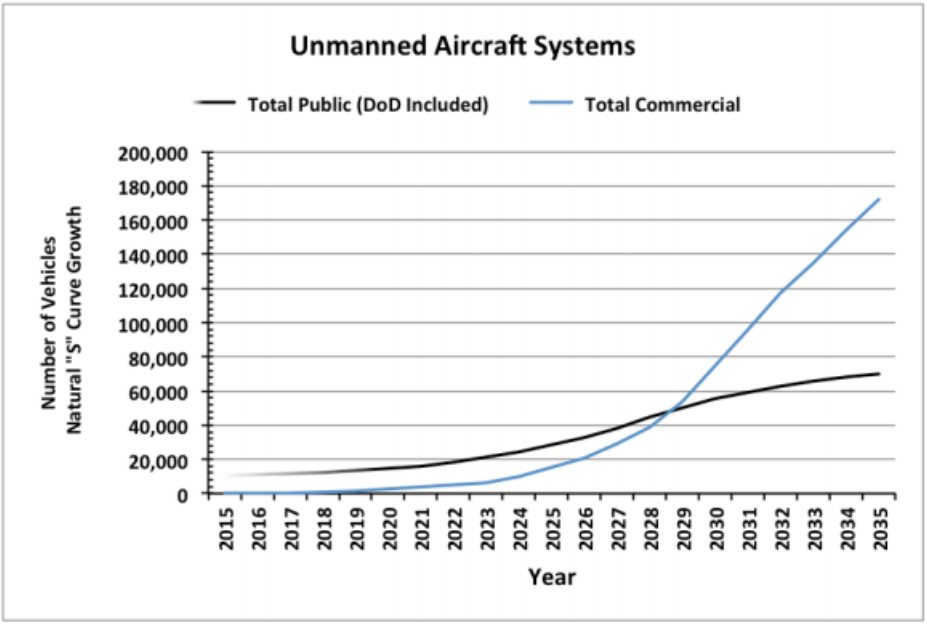

Figura 1.1: Predicción del crecimiento de UAVs [11]

Teniendo en cuenta lo anterior, el presente Proyecto de Fin de Grado aborda esta corriente, diseñando y desarrollando un sistema capaz de proveer seguridad a nivel de datos, permitiendo la transmisión de información por canales poco fiables.

1.2. Objetivos y Alcance del Proyecto

La finalidad de este proyecto es el diseño y desarrollo de un sistema para la entrega segura de información en entornos de UAVs. El objetivo principal se desglosa en los subobjetivos siguientes .

- Estudiar los mecanismos de seguridad actuales para proteger comunicaciones en red, así como los nuevos mecanismos emergentes de seguridad basados en contenido que se están desarrollando en los nuevos modelos de Redes Centradas en la Información o ICN. Las ICN proponen un enfoque distinto al tradicional de Internet, enfatizando el acceso a los contenidos independientemente de su localización. Su idea clave es usar nombres para proveer de dirección a los contenidos y para reenviar el contenido en la red.
- Diseñar y desarrollar una Application Programming Interface (API) que permita entregar información a múltiples destinatarios de manera segura, verificando la autenticidad de los datos distribuidos.
- Diseñar y desarrollar un conjunto de mecanismos de control que permitan solicitar la información a un equipo conectado a la red que disponga de esa información.

Estos mecanismos estarán basados en el protocolo Session Initiation Protocol (SIP) [8], estandarizado por la comunidad internacional Internet Engineering Task Force (IETF) para el control de comunicaciones multimedia.

- Validar el desarrollo realizado y su viabilidad en un entorno de laboratorio, concretamente en el laboratorio del Departamento de Telemática de la Universidad Carlos III de Madrid.

1.3. Estructura del documento

El presente documento está dividido en siete capítulos y tres apéndices, los cuales están descritos a continuación:

- El capítulo 1, el actual, es la **Introducción**, el cual desarrolla el contexto del proyecto, la motivación del mismo, y la organización del documento.

- El capítulo 2, **Estado del arte**, desarrolla la historia de los UAVs, así como nuevos equipamientos para transmitir datos, introducción a la seguridad que va a ser añadida a los datos, protocolos para comunicaciones multimedia. También se introducirá el marco regulatorio de los UAV.

- El capítulo 3, **Diseño**, expone el escenario del problema y los requisitos que debe cumplir la API, así como los diseños de los diferentes módulos que toman partido en la comunicación.

- El capítulo 4, **Implementación**, ofrece una explicación detallada de la solución propuesta en el capítulo 3.

- El capítulo 5, **Validación y Pruebas**, lista las diferentes pruebas efectuadas para la comprobación del rendimiento y correcto funcionamiento de la API.

- El capítulo 6, **Gestión del Proyecto**, capítulo donde se recogen las tareas y horas empleadas, además de un análisis económico sobre el proyecto.
- El capítulo 7, **Conclusión y Líneas Futuras**, en este capítulo se comenta los resultados obtenidos y los problemas encontrados en el desarrollo de la API. Además, se introducen futuros trabajos y mejoras de la API.

- Apéndice A, **Introduction**, en este capítulo se encuentra la traducción al inglés del primer capítulo **Introducción**, como requisito de la redacción del Trabajo de Fin de Grado.
- Apéndice B, **Conclussions and Future Lines Work**, este capítulo está compuesto por la parte traducida al inglés del capítulo 7, **Conclusión y Trabajo Futuro**.
- Apéndice C, **Extended Summary**, se presenta un resumen extendido de la memoria del proyecto en inglés.

Capítulo 2

Estado del Arte

2.1. UAV

Una visión simplificada de un Vehículo Aéreo No Tripulado, VANT en sus siglas en español o UAV en sus siglas en inglés, es un avión sin tripulación a bordo (recientemente se les ha comenzado a denominar Remotely Piloted Aircraft Systems (RPAs)), y una conexión radio, al final de la cual se encuentra la tripulación terrestre, que controla el UAV. El conjunto completo de UAV se le denomina Unmanned Aircraft Systems (UAS) y está compuesto por una Estación de Control o GCS que contiene los operadores de control; un sistema de comunicaciones (envía entradas de control al UAV y éste devuelve la carga y otros datos al GCS); y por último, el equipamiento de apoyo (se pueden encontrar: elementos de telemetría, sistemas de vídeo y audio), y transporte de objetos. Un UAS puede contener una GCS y uno o varios UAVs.

Los UAVs son comúnmente utilizados tanto civilmente como militarmente. Por ejemplo, en el ámbito civil se usa para la fotografía aérea, sector de la agricultura, conservación del medioambiente, mantenimiento en compañías eléctricas y cualquier aplicación imaginable. En el ámbito militar, está orientado al seguimiento de flotas enemigas, retransmisión de señales de radio, reconocimiento, localización de campos de minas, sistema de interferencia de radares y muchas más opciones.

La diferencia entre aviones de aeromodelismo, drones y UAVs, es que los aviones de aeromodelismo son usados sólo para ocio y siempre tienen que estar a la vista del operador, mientras que los drones pueden ser manejados sin estar a la vista del operador, pero su capacidad de computación o inteligencia es moderada (despegan con una misión pre-programada, con un vuelo pre-programado y vuelta a la base). Los UAV son capaces de tomar acciones correctivas y/o alertar a su operador para el evento [1].

Los UAVs pueden ser categorizados dependiendo de su tamaño, donde se pueden encontrar [2]:

[1] Estas definiciones pueden ser encontradas en [22], sección 1.2

[2] Según [22]

– *HALE*, High Altitude Long Endurance. Pueden alcanzar más de 15 kilómetros de altitud y tener una duración de vuelo de 24 horas. Llevan a cabo reconocimientos y vigilancias globales. Son operados desde bases fijas.

– *MALE*, Medium Altitude Long Endurance. Trabajan entre los 5 y 15 kilómetros de altura y duran 24 horas. Desempeñan un rol similar a los sistemas *HALE*. También son operados desde bases fijas.

– *TUAV*, Tactical UAV. Trabaja en un rango del orden entre 100 y 300 kilómetros. Son de menor tamaño que los *HALE* o *MALE* y también son operados desde tierra.

– *Close-Range UAV*, usados por cuerpos móviles de batalla, operaciones navales u otros propósitos civiles. Normalmente, operan en rangos de 100 kilómetros y son los que más usos prolíficos tienen en ambos campos, incluyendo roles muy diversos, como seguridad aérea, inspección de tendido eléctrico, etc.

– *Mini UAV*, se relaciona con la masa (probablemente menor a 20 kilogramos), capaz de ser lanzado desde las manos y operar en rangos hasta 30 kilómetros.

Cada UAV está diseñado para una aplicación particular. Una profundización de las aplicaciones más comunes que los UAVs pueden desempeñar son [22]:

- *Rol medioambiental*. En operaciones de monitorización de catástrofes ambientales como la contaminación química o nuclear que pondrían a la tripulación a bordo en riesgo. También son usados en fumigaciones con químicos tóxicos.

- *Rol peligrosidad*. Reconocimiento de un campo enemigo altamente defendido, ya que gracias a su sigilo es más difícil detectar y difícil de derribar. Los operadores de los UAVs aún estando en el campo de batalla, no están bajo peligro directo por derribo, por lo que pueden tomar decisiones más efectivas. En el ámbito civil, son usados para la inspección de tendido eléctrico o el control de fuego de un bosque.

- *Rol de investigación*. Los UAVs están siendo empleados para investigaciones y desarrollos en el campo aeronáutico. Con el propósito de pruebas, los UAVs son usados como réplicas de vehículos aéreos tripulados para hacer pruebas en el aire en condiciones reales, más baratas y con menor riesgo.

- *Impacto medioambiental*. Un UAV normalmente causa menor impacto medioambiental o de contaminación que un avión tripulado haciendo la misma tarea. Al ser menor en tamaño y masa, consume menos, produciendo menores niveles de emisión de gases nocivos y ruido.

- *Razón económica*. Los UAVs son generalmente más pequeños que los aviones tripulados que ejecutan los mismos roles, y son considerablemente más baratos. El coste de operación también es menor ya que tiene pocos costes de mantenimiento, combustible y almacenamiento. Los costes laborales y seguro de los operarios son normalmente más baratos.

En el entorno de **comunicaciones**, el principal requisito es conseguir un enlace de datos de subida y bajada entre GCS y el avión. El medio de transmisión es normalmente mediante radio-comunicación en línea de vista, pero existen alternativas usando haz de láser.

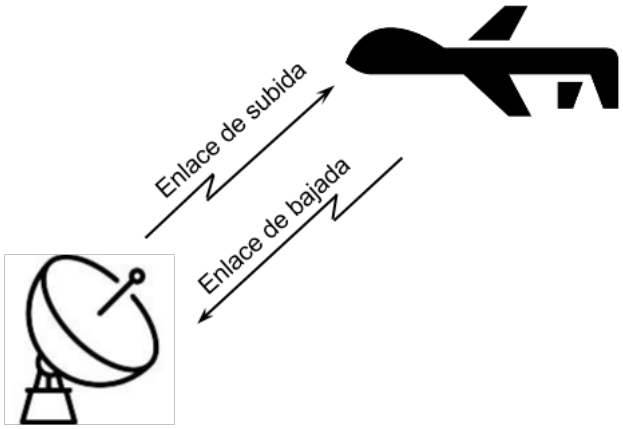

Figura 2.1: Enlaces de Comunicación

Enlace de subida (Uplink):
Es la parte de la comunicación dónde el GCS transmite datos al UAV. Entre estos datos se encuentran las órdenes del plan del vuelo (que se almacenan en el Automatic Flight Control System (AFCS)), la transmisión en tiempo real de los comandos de control de vuelo al AFCS, cuando es requerida la interacción humana (cambio de las órdenes establecidas anteriormente en el plan de vuelo); datos de control de los equipos de apoyo del avión (por ejemplo, la modificación del foco de las cámaras infrarrojas).

Enlace de bajada (Downlink):
Parte de la comunicación encargada de la transmisión de datos desde el UAV hacia el GCS. Entre estos datos transmitidos se encuentran la información sobre la posición del UAV, la transmisión de las imágenes originadas por los equipos de apoyo del UAV (como por ejemplo las imágenes captadas de las cámaras infrarrojas) y el envío de la información telemétrica del UAV (como la temperatura del motor, altitud, nivel de combustible, etc.).

El nivel de potencia eléctrica, complejidad de procesamiento y el diseño de la antena, el peso y coste de la comunicación radio vendrán determinados por:

1. El rango de operación del vehículo aéreo desde la estación de transmisión.
2. La sofisticación demandada por la transmisión de bajada de la información de la carga y del los parámetros del UAV.
3. La necesidad de seguridad.

Todos estos subsistemas necesitan trabajar juntos para conseguir la ejecución del sistema completo. Por tanto, todos los componentes de comunicación del UAS deben interactuar con los mismos protocolos (o protocolos compatibles), ya que deben "hablar el mismo idioma"para poder trabajar en conjunto.

Debido a que los UAS pueden ser adquiridos y desplegados en diferentes países, éstos deben adecuarse a la legalidad de dicho país con respecto a los UAVs, además de utilizar las frecuencias provistas por parte de los gobiernos para evitar interferencias.

2.2. Seguridad

La seguridad es un elemento fundamental en el traspaso de información desde tiempos muy remotos. En esta sección se expondrán las diferentes épocas en las que se divide la historia de la criptografía, así como los diferentes algoritmos utilizados en el presente. También incluye un apartado detallado sobre firmas digitales, necesario para la comprensión del presente Trabajo Fin de Grado.

2.2.1. Introducción a la Criptografía

La Criptografía es la ciencia que usa las matemáticas para convertir un Texto plano en un texto cifrado, y viceversa, de un texto cifrado convertirlo en un Texto plano con el uso del descifrado. La Criptografía permite almacenar datos con necesidad de protección que no pretenden ser leídos por agentes externos a la conversación. El nivel de seguridad proporcionado con esta combinación depende de la robustez del algoritmo criptográfico y la seguridad de la clave.

Con el paso del tiempo, se puede dividir la historia de la criptografía en dos grandes períodos, la criptografía clásica y la criptografía moderna.

2.2.2. Criptografía clásica

El ser humano siempre ha sido muy reticente respecto a sus secretos, por eso ha buscado mecanismos para mantenerlos a salvo de personas externas a esa comunicación. Julio César utilizaba una técnica sencilla para transmitir las órdenes a sus altos mandos militares sobre estrategias. Esta técnica usaba Cifrados de sustitución, se basaban en sumar un número al número de orden de cada letra. De esta manera si se quería mandar el mensaje *HOLA* se transformaría en un nuevo mensaje *KROD*, por lo que el general que recibiese ese mensaje alterado sabía que tenía que restar tres posiciones a cada letra para conseguir el mensaje original. Este método en particular es un tipo de *Cifrado monoalfabético*.

El cifrado evolucionó del monoalfabético a múltiples letras, entre los que se encuentran el cifrado *Playfair* que cifra de dos en dos letras; cifrado *polialfabético*, que consiste en la sustitución de símbolos en función de su posición en el texto; el cifrado de *Vigenère*, el cual usa como alfabeto las 26 permutaciones del alfabeto formando una tabla; el cifrado de *Vernam*, que ejecuta un Exclusive OR (XOR) del texto plano con una clave cíclica o presudo-aleatoria de la misma longitud. Este tipo de mecanismos de cifrado se puede criptoanalizar efectuando un estudio estadístico sobre la frecuencia de aparación de pares y tripletas de símbolos en el lenguaje en que esté escrito el texto claro [19].

Pero donde la criptografía clásica alcanzó su mayor desarrollo fue con el invento de las máquinas de rotores. La máquina de rotor que más impacto ha tenido en nuestra historia bélica se la conoce como la *Máquina Enigma* . Esta máquina fue fabricada en 1923 por un ingeniero alemán para usos civiles, pensada para la facilitación de las comunicaciones seguras. Su similitud con una máquina de escribir permitía que personas que deseasen codificar un texto, sólo tuviesen que teclearlo y los símbolos pertenecientes al texto Cifrado se irían iluminando en un panel. El receptor copiaba el texto Cifrado en su máquina

y el texto original aparecía. La clave perteneciente al Cifrado estaba formada por tres rotores, que daban nombre al tipo de máquina, que dependiendo de su posición el Cifrado cambia. Cada vez que se pulsa una tecla el primer rotor avanza una posición, mientras que el segundo avanza cuando el primer rotor ha dado una vuelta completa y el tercer rotor avanza una posición cuando el segundo rotor ha dado una vuelta completa.

Figura 2.2: Imagen de la máquina Enigma [1]

2.2.3. Criptografía moderna

La criptografía moderna está orientada a la seguridad en sistemas informáticos. El objetivo de estos sistemas es preservar la información frente a alteraciones, ya sean de carácter software como hardware. Además de evitar estas alteraciones hay que eludir los accesos no autorizados a nuestro sistema. Además de estos requisitos es necesario que la información que contiene el sistema esté accesible siempre que sea necesario. Las cuestiones de seguridad más relevantes son las siguientes:

- **Seguridad física**. Seguridad orientada a la salvaguarda de los soportes físicos de la información, es decir, prevención de ataques físicos, medidas contra incendios y sobrecargas eléctricas. Esta medida sigue una política de copias de respaldo que permite mantener disponible la información en otro soporte físico si el primero ha sido dañado.
- **Seguridad en el canal de comunicación**. Los canales de comunicación suelen pertenecer a terceros, por ese motivo estos canales no se definen seguros, ya que pueden estar siendo escuchados o intervenidos, poniendo en riesgo la información que se transmite. Por lo cual se tendrá que establecer mecanismos de seguridad para garantizar la integridad de la información transmitida por dichos canales.
- **Control de acceso**. Los datos deben ser accesibles en cualquier momento, pero esos datos sólo deben estar disponibles para agentes autorizados para usar la información. Se necesita establecer privilegios individualizados dependiendo del nivel

de seguridad de los distintos datos almacenados. Además de este control de acceso, la información debe ser preservada mediante cifrado en los dispositivos de almacenamiento.

- **Autenticación**. Se necesita verificar la autenticidad de los agentes que acceden a la información ya que no se puede permitir que su identidad no ha sido suplantada. Además, hay que comprobar que los datos recibidos y enviados son auténticos y no han sido alterados. Todas estas verificaciones se efectúan en el entorno de autenticación.
- **No repudio**. Al recibir unos datos, además de verificar que no han sido comprometidos, así como la identidad de los agentes, es necesario que el emisor no pueda *repudiar* los datos enviados por él, asumiendo su autoría.
- **Anonimato**. Proceso contrario al *no repudio*. En determinados contextos es necesario la conservación del anonimato para poder preservar su identidad. Esta cuestión es muy difícil de cumplir, ya que al conectarse a una red se deja un rastro de la identidad o la posición del usuario.

La criptografía moderna se puede dividir dependiendo del uso de la clave. La clave simétrica utiliza sólo una clave para el proceso de cifrado y descifrado que tiene que ser compartida tanto como por el emisor como por el receptor; mientras que la clave pública o asimétrica genera dos claves, una para cifrar y otra para descifrar. Los procedimientos utilizados por ambas son:

Clave Secreta o Simétrica

Genera una única clave para el proceso de cifrado y descifrado, haciendo casi imposible el envío de esa clave al receptor de una manera segura.

- **Cifrado simétrico**. Se basa en la confunsión y difusión de la información. La confunsión consiste en tratar de ocultar la relación existente entre el texto plano, el texto cifrado y la firma. La difusión trata de repartir la influencia de cada bit del mensaje original entre el mensaje cifrado. [19]. La mayoría de los algoritmos de cifrado simétrico utilizan un cifrado de bloque.

    - **Cifrado de bloque.** El procedimiento seguido por una mayoría de Cifrados simétricos es dividir el mensaje a cifrar en bloques de tamaño fijo y, dependiendo del algoritmo usado, aplicar una serie de reglas para transformar cada uno de los bloques en Texto plano formando un mensaje con texto Cifrado. Si en la división del mensaje en bloques, un bloque no alcanza el tamaño fijo, se le añade Padding. Entre los algoritmos más importantes se encuentran:
        - **Data Encryption Standard (DES)**. Codifica bloques de 64 bits utilizando claves de 56 bits. Es una red de Féistel de 16 rondas más una permutación que se aplica al inicio y otra permutación que se aplica al final.
        - **International Data Encryption Algorithm (IDEA)**. Utiliza bloques de 64 bits de longitud y emplea una clave de 128 bits. Al igual que DES, utiliza el mismo algoritmo para cifrar que para descifrar.

- **Advanced Encryption Standard (AES)**. Más conocido como el algoritmo *Rijndael*, es el estandard actual para el empleo en aplicaciones criptográficas no militares. AES está diseñado para manejar longitudes de clave y bloque variables comprendidas entre los 128 y 256 bits.

- **Cifrado de flujo.** Orientados al Cifrado de mensajes de longitud arbitraria. El procedimiento que sigue este tipo de Cifrado es tomar de forma continua la ristra de bits del mensaje y hacer un XOR. No es un sistema perfecto, ya que depende del valor inicial de la semilla.

- **Intercambio seguro de clave**. Cuando el emisor y el receptor no comparte la misma posición física es necesario el envío de la clave para poder cifrar y descifrar. Por lo que es necesario un canal de distribución de claves suficientemente seguro para que no sea interceptada. Los métodos pensados para esta distribución son:

  – *Envío Directo*, se genera una clave de sesión que va a ser compartida por los extremos, pero si se intercepta esta clave, un agente externo puede suplantar a uno de los extremos y el otro no lo notaría. Para evitarlo, se añade una marca de tiempo para que las sesiones expiren, aún con esto, si existen más de dos nodos, el número de claves incrementaría haciéndose casi imposible el manejo.

  – *Key Distribution Center (*KDC*)* , opción que implica a un tercero de confianza o KDC, el cuál tendrá todas las claves y proveerá a los nodos cuando éstos hagan peticiones de ellas para asegurar un canal de distribución con una clave de sesión generada por el KDC. Tal como se muestra en la figura 2.3

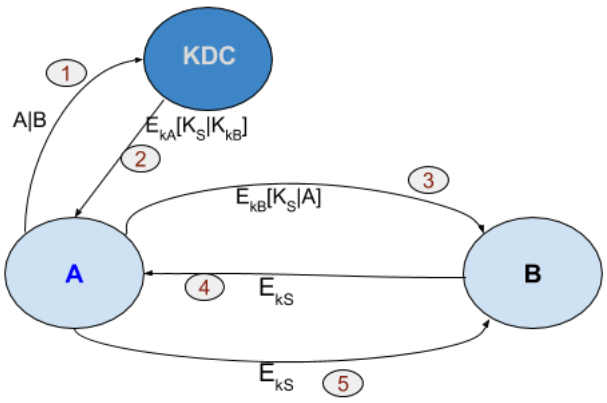

Figura 2.3: Key Distribution Center

Con esta aproximación sólo se obtiene una solución parcial, ya que si se logra romper la clave de sesión, un agente externo puede suplantar al nodo A. Por lo cual, es necesario la introducción de la Criptografía asimétrica

- **Autenticación e integridad en criptografía simétrica**. Los Message Authentication Code (MAC) se caracterizan por el empleo de una clave secreta para poder calcular la integridad del mensaje. Dado que la clave sólo es conocida por el emisor y el receptor, este último es el único que puede comprobar la integridad del mensaje mediante los cálculos de la función correspondiente [19].

Clave Pública o Asimétrica

La gran diferencia con respecto a la clave secreta o simétrica, es que con el Cifrado asimétrico no se genera una clave única, sino que se generan dos claves, una pública para cifrar y una privada para descifrar. El rol que suele desempeñar los algoritmos asimétricos es el cifrado de la clave de sesión, simétrica, de cada mensaje o transacción, sin necesidad de transmitir la clave de decodificación, por lo que su uso es factible en canales inseguros [19].

En la figura 2.4, se puede apreciar como el equipo A solicita a B el envío de datos. El equipo B genera el par de claves (pública y secreta), enviando su clave pública al equipo A. Una vez que el equipo A recibe la clave pública de B, cifra el mensaje original con la clave pública, y se lo envía a B, que puede descifrarlo con su clave privada.

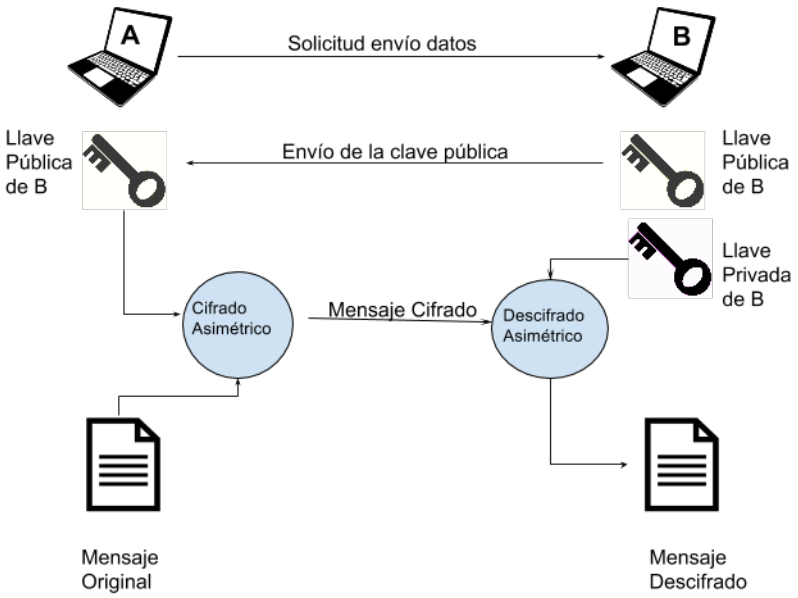

Figura 2.4: Cifrado Asimétrico

Entre los algoritmos más importantes del cifrado asimétrico se encuentran:

- **Diffie-Hellman.** Algoritmo propuesto para solventar el problema del acuerdo e intercambio de claves, pero no puede ser usado para cifrar o descifrar. Este algoritmo está basado en principios matemáticos.[12]

- **El Gamal.** En sus orígenes fue diseñado para encriptar mensajes en una manera matemática, pero luego se descubrió su utilidad en el proceso de firma, creando una base matemática utilizada en las firmas modernas. [23]

- **Rivest, Shamir and Adleman (RSA).** Este algoritmo sirve tanto como cifrador como autenticador. Con el algoritmo RSA primero se Firma y luego se cifra, ya que al contrario existen métodos que permiten manipular el mensaje. Está basado en la dificultad de factorizar números grandes. Primero, se eligen dos números primos grandes, p y q, después se calcula el producto de ambos, $n = pq$. Ahora se elige un primo relativo [3] e con $(p-1)(q-1)$. La clave pública estará formada por (e, n). e tiene que tener inversa del $mod(p-1)(q-1)$, denominando a esta inversa d. La clave privada será (d, n). El proceso de cifrado se genera con la clave pública: $c = m^e * mod(n)$, mientras que el descifrado se genera con la clave privada: $m = c^d mod(n)$, siendo c el texto cifrado, y m el texto plano.

2.2.4. Firmas Digitales

Las Firmas digitales permiten al destinatario de la información verificar la autenticidad del origen de la información, además de que la información esté intacta. La firma digital provee *autenticación* e *integridad de los datos*, además del *no repudio*.

El propósito de las Firmas digitales es análogo a una Firma convencional (escrita a mano), con la diferencia de que si la Firma digital se modifica, también se modifica el contenido firmado y la identidad del origen.

El proceso de creación de una Firma es, teniendo el Texto plano, cifrar con la clave privada del origen, solamente en posesión del originario, y verificar con la clave pública del originario de la Firma que puede estar en posesión de cualquier persona. [2]

Funciones Resumen

Las funciones resumen, o "Hash Functions" en inglés, nacieron de la necesidad de reducir el tamaño de los mensajes que iban a ser Cifrados para que la clave resultante de los procesos descritos en los apartados anteriores no fuese demasiado grande. Estas funciones toman un mensaje de cualquier tamaño y lo transforma en otro mensaje completamente distinto con una longitud mucho menor [2]. Entre las funciones se encuentran un tipo especial que emplean una clave adicional para añadir seguridad, llamada MAC. Las funciones que no usan una clave adicional, son denominadas Modification Detection Codes (MDC).

Para que un mensaje sea autenticado, primero se le aplica una función resumen y posteriormente se cifra con la clave privada. $E_{Kp}(r(m))$, donde E_{Kp} representa a la codificación con clave pública, y $r(m)$ es la función resumen. $r(m)$ tiene una longitud fija independiente de la longitud del mensaje. Dado m es sencillo calcular $r(m)$, pero imposible el proceso inverso $(r(m) \rightarrow m)$. Es computacionalmente imposible obtener un mensaje cuya función resumen sea igual a la función resumen del mensaje original. [19]

[3] Dos números naturales son primos relativos si su máximo común divisor es 1.

Dentro de las funciones resumen más usadas están:

- **Message-Digest Algorithm 5 (MD5)** Es uno de los algoritmos más populares de generación de firmas. En primer lugar se alarga el mensaje hasta que su longitud sea de 64 bits inferior a un múltiplo de 512. El alargamiento se lleva a cabo añadiendo un 1 seguido de tantos ceros como sea necesario. Después, se añaden 64 bits con el valor de la longitud original del mensaje, empezando por el menos significativo. De esta forma se tiene un mensaje como un número entero de 512 bloques, además de tener información sobre su longitud.

- **Secure Hash Algorithm (SHA)** Este algoritmo tiene el adjetivo de seguro porque es computacionalmente inviable encontrar un mensaje el cual corresponda con un resumen de mensaje dado, o encontrar dos mensajes diferentes que produzcan el mismo resumen. Cualquier variación en el mensaje producirá un resumen totalmente distinto el cual hará fallar la verificación de la Firma.

  SHA obtiene una representación condensada de un mensaje o de un fichero, siempre y cuando el tamaño no exceda de 2^{64} bits, SHA producirá un mensaje de 160 bits. Esta salida puede ser usada como entrada en un algoritmo de Firma que genera o verifica la Firma del mensaje. Firmando con el resumen se consigue un tamaño mucho menor que usando el mensaje original, lo que se traduce como eficiencia del proceso. La misma función resumen debe ser usada por el verificador como fue usado por el generador de la Firma. [10]

  Dentro del algoritmo resumen SHA podemos encontrar varios tipos:

  - **SHA-1**. Produce mensajes de 160 bits, a partir de bloques de 512 bits del mensaje original.
  - **SHA-2**. Produce mensajes de hasta 256 bits, a partir de bloques de hasta 1024 bits.
  - **SHA-3**. Produce mensajes hasta 512 bits de longitud, con bloques de longitudes desde 576 bits hasta 1152 bits.

2.3. ICN

ICN es una nueva propuesta de la comunidad investigadora para reemplazar Internet, ya que enfoca el acceso al contenido independientemente de su localización, al contrario que el enfoque tradicional de Internet que está basado en la identificación y localización de los dispositivos finales. El concepto novedoso de las ICNs es el uso de nombres para direccionar los contenidos, el nombramiento del enrutamiento, seguridad directamente aplicada a los contenidos y el uso de memoria caché dentro de la red.

Estos conceptos permiten desplegar una arquitectura más eficiente para la distribución de contenido, evitando el uso de "parches" en la arquitectura de Internet tales como Internet Protocol (IP) Multicast, Domain Name System (DNS) e Internet Protocol Security (IPSec). De forma nativa, las ICNs proveen nuevas funcionalidades como mecanismos para aumentar la disponibilidad del contenido, soporte a la seguridad del contenido y soporte a la movilidad.

Una solución de ICN es una arquitectura Content-Centric Networking (CCN), con unas características principales como es la división del contenido en trozos, nombrar objetos con un identificador único y jerárquico, el cual puede ser solicitado individualmente. Los nombres de CCN están compuestos por un número arbitrario de octetos, transparente a la capa de protocolo, y puede incluso estar cifrado.

La estructura de la arquitectura de CCN está dividida en dos módulos básicos: una declaración de interés por un trozo de contenido específico y la transmisión de este trozo en respuesta al interés. Los nodos envían el paquete de datos en respuesta al paquete de interés en caso de que ellos tengan el trozo de contenido solicitado almacenado localmente. En el caso de que no tenga el paquete solicitado, el paquete de interés es reenviado a sus vecinos hasta que alcance el nodo que tiene almacenado el contenido en caché. Los datos sólo son enviados en respuesta a una solicitud. [17]

Nuestra propuesta se basa en conceptos de ICN, como solicitar la recepción de información, siendo el transmisor de esta información cualquier usuario que disponga de ella. Como en ICN, la API a diseñar enfoca la seguridad a nivel del contenido, y no a la seguridad de los canales de transmisión.

2.4. SIP y SDP

Origen de la transmisión de voz/datos

La red telefónica alcanza casi todos los países del mundo, todos usan una red de *conmutación de circuitos* para comunicarse entre ellos. La red telefónica es una red de alta fiabilidad. Las llamadas entre dos teléfonos son enrutados primero al intercambiador local del destinatario de la llamada, el cual puede ser anunciado de la llamada mediante el zumbido de su teléfono. Además de la transmisión de voz, también pueden ser usados para transportar diferentes tipos de tráfico entre ordenadores y señales de control entre terminales.

Ante la evolución de la tecnología de analógico a digital, los conmutadores de circuitos evolucionaron a *conmutadores de paquetes*, y de Frequency Division Multiplexing (FDM) a Time Division Multiplexing (TDM), ya que se aumenta considerablemente la eficiencia. Usando los circuitos digitales se puede hacer uso de datagramas, que necesitan tablas de enrutamiento en las redes para alcanzar su destino, y uso de circuitos virtuales, que almacenan el estado del circuito e identidades, para emparejar paquetes entrantes con el circuito virtual al que pertenece. [13]

Transmisión a través de Internet

En Internet se utliza *conectividad* IP, implementada para la transmisión de ficheros, servicios de correo electrónico, etc. Este Protocolo está implementado sobre la capa de red. Los sistemas en el extremo de la conexión tienen la tarea de controlar el tráfico de la red IP, incluidos control de flujo y detección de errores. La red no envía ninguna notificación al sistema final indicando si el paquete ha sido o no entregado, a esto se le define como Best Effort. [9]

Para utilizar una red de conmutación de paquetes por Internet se desarrolló el protocolo de transporte User Datagram Protocol (UDP), el cual asume que IP es usado como Protocolo subyacente. Este Protocolo está orientado a la transacción no a la conexión, y por tanto, la entrega y la protección no están garantizadas. Debido a esto, es necesario el uso de protocolos como SIP que actúen de señalización en la comunicación entre dos equipos, así el equipo receptor tiene constancia de que su petición de datos ha llegado al emisor.

2.4.1. SIP

SIP es un protocolo de la capa de aplicación del modelo TCP\IP, diseñado para la creación, modificación y terminación de sesiones con uno o más participantes. Estas sesiones pueden ser llamadas sobre Voice Over Internet Protocol (VoIP) y distribución multimedia. El protocolo SIP es un protocolo de los denominados peer to peer, es decir, cada una de las partes que intervienen en la comunicación puede hacer la función de cliente o de servidor.

Las invitaciones de SIP son usadas para transportar descripciones sobre la sesión, normalmente escrito en formato Session Description Protocol (SDP), lo que permite a los participantes ponerse de acuerdo con un conjunto de medios de comunicación. Estas invitaciones solicitan al llamado a unirse a una conferencia o establecer una conversación uno a uno.

Los distintos elementos de SIP son:

- **Agentes de usuario**. Son los extremos de la sesión, es decir, los terminales que emiten y consumen los mensajes del protocolo SIP.
- **Servidores de registro**. El agente de usuario al iniciar envía una petición *REGISTER* a un servidor de registro, informándole de la dirección lógica del usuario.
- **Proxys y servidores de redirección**. Los proxys se encargan de encaminar los mensajes entre el cliente y el servidor, mientras que los servidores de redirección hacen peticiones DNS para localizar los dominios (direcciones de los usuarios).

Cuando uno de los usuarios quiere abandonar la conversación, únicamente tiene que señalizar su propósito con una solicitud de salida, para que el servidor SIP no entregue más paquetes a ese destino. Gracias a esto la eficiencia del ancho de banda usando SIP es máxima, ya que no se desperdicia con conexiones sin usuario.

Los distintos mensajes de señalización de SIP se pueden dividir en dos grupos: solicitudes y respuestas. Cada uno de estos grupos tienen sus propios mensajes [13]:

- **Solicitudes**
  - **INVITE**, encargado de enviar la solicitud de comienzo de conversación. Envía los tipos de codificación soportados por el emisor y la identificación de la llamada.
  - **ACK**, mensaje que informa el fin de la negociación sobre la codificación que será usada en la comunicación, precedente a la transmisión de datos.

- **BYE**, el usuario informa a los integrantes de la conversación que abandona la sesión.
- **REGISTER**, mensaje que el usuario envía al servidor SIP para registrarse, notificando su posición.
- **CANCEL**, mensaje que cancela una solicitud. Si la solicitud ha llegado al destino, este mensaje no tendrá efecto.
- **OPTIONS**, mensaje que indica sobre las capacidades de transmisión del usuario.

- **Respuestas**
  - **1XX**: mensaje de información.
  - **2XX**: mensaje de éxito.
  - **3XX**: mensaje de redirección.
  - **4XX**: mensaje de error de cliente.
  - **5XX**: mensaje de error de servidor.
  - **6XX**: mensaje de fallo global.

En la figura 2.5 se muestra un diagrama de intercambio de mensajes SIP:

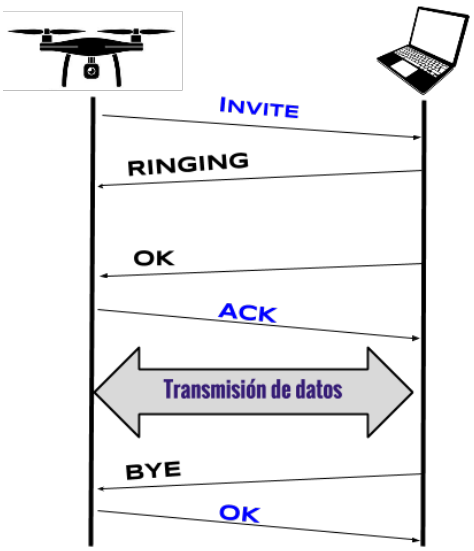

Figura 2.5: Establecimiento y terminación de sesión en SIP

2.4.2. SDP

Una descripción de SDP contiene información del nivel de sesión y de medio de comunicación. Por ejemplo, la información de sesión se refiere al usuario originario de la sesión o el nombre de la sesión. La información sobre el medio de comunicación se aplica a un determinado flujo, como puede ser el códec usado para codificar el flujo de audio o el número de puerto donde se dirige el flujo de vídeo. Esta descripción es incluida en el protocolo SIP. [7]

El cuerpo de la carga de SDP es:

v =	Tipo de versión SDP
o =	Creador de la sesión y dirección IP de su localización
s =	Nombre de la sesión
i =	Información general sobre la sesión
u =	URL donde se puede conseguir más información sobre la sesión
e =	Dirección e-mail de la persona de contacto para esta sesión
c =	Dirección multicast donde puede ser recibida la sesión
t =	Tiempo durante que la sesión ha estado activa
a =	Informa si la sesión es interactiva o si es solo de recepción
m =	Tipo de medio (audio, vídeo, audio + vídeo)

2.5. Marco Regulatorio sobre UAVs

El gobierno de España en el año 2014 mediante un decreto ley [14], aprobó una serie de medidas temporales para la regulación del uso de UAVs. Este decreto ley hace diferencia entre dos tipos de UAVs, los de peso inferior a 150kg seguirán la norma de la Agencia Estatal de Seguridad Aérea (AESA), y los de peso superior a 150 kg deberán remitirse a la normativa de la European Aviation Safety Agency (EASA). Esta normativa está presente para el uso de UAVs con fines comerciales; en cambio, para uso con fin lúdico o deportivo esta norma no está presente, ya que en este uso no es legalmente un UAV sino un vehículo de radiocontrol. Entre las normas para UAVs con peso menor de 150 kg se encuentran:

- **Peso menor a 2kg.**

    1. No será necesario la inscripción en el registro de aeronaves, así como no será necesario disponer de un certificado de aeronavegabilidad.
    2. Tanto la permitividad respecto al alcance visual como a la distancia máxima de vuelo, están delimitados por la estación de control.
    3. La altura máxima permitida es de 120 metros.
    4. Necesario la notificación de aviso de uso de UAV

- **< 25kg**
    1. Necesario en el registro de aeronaves, disponer de un certificado de aeronavegabilidad, así como un carnet de piloto de drones.
    2. El alcance visual permitido es el campo de visión del piloto, con una distancia máxima de 500 metros.
    3. La altura máxima permitida es de 120 metros.
    4. Es necesaria una declaración de responsabilidad al hacer uso de este vehículo.
- **> 25kg**
    1. Necesario registrar el vehículo en el registro de aeronaves, disponer de un certificado de aeronavegabilidad, además de disponer de un carnet de piloto de drones.
    2. Tanto el alcance visual como la distancia y altura máximas, están limitados según el certificado de aeronavegabilidad.
    3. Es necesaria una placa de identificación así como una matrícula del vehículo.
    4. Se requiere de una autorización por parte de AESA y el registro de la matrícula para poder volar el vehículo.
- **Para todos los UAVs**
    1. El dispositivo es requerido de una placa identificativa con los datos del fabricante y la empresa que lleve a cabo las operaciones.
    2. Está prohibido sobrevolar núcleos urbanos o espacios de alta densidad de población sin un consentimiento previo de la AESA.

En este decreto no se especifica la restricción en seguridad sobre los datos transmitidos desde ningún tipo de UAV. Debido a ello, este proyecto se va a centrar en un mecanismo que permita añadir una seguridad a nivel de contenido.

Capítulo 3

Diseño del sistema

En este capítulo se expone el problema y la solución que este proyecto propone. Además de cuáles van a ser los pasos necesarios que debe tomar la API para conseguir el objetivo del proyecto.

3.1. Escenario del problema

Tal como se ha comentado anteriormente, el principal objetivo de este proyecto de Fin de Grado es el diseño y la implementación de un sistema para la distribución segura de información en entornos de UAVs. Este sistema se basará en el concepto de la seguridad centrada en la información, desarrollada en el ámbito de las ICN.

El sistema diseñado permitirá a un usuario interesado solicitar y recibir la información de telemetría de un UAV específico, y autenticar el origen de la información recibida. El sistema también permitirá solicitar la información de telemetría a cualquier equipo receptor, el cual la retransmitirá al solicitante que podrá autenticar el origen apropiado de la misma, es decir, que proviene del UAV.

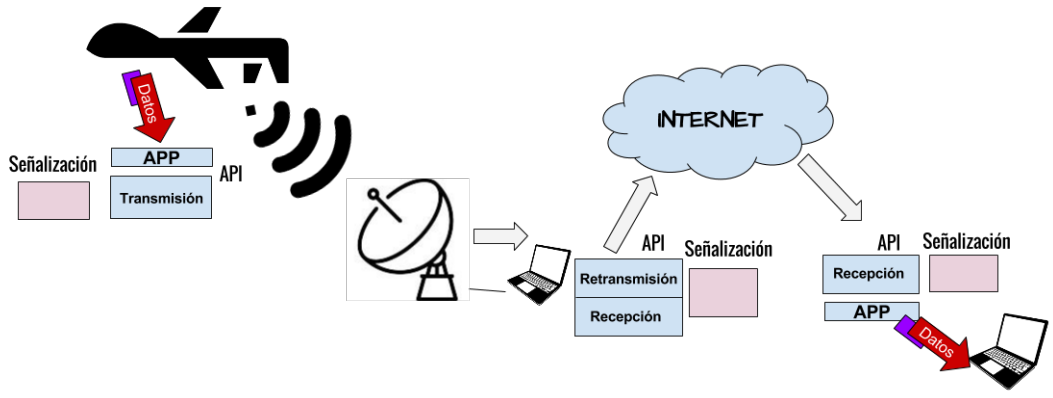

Figura 3.1: Escenario del problema

El diseño propuesto tiene dos componentes principales: el módulo de señalización, encargado de solicitar el envío del contenido multimedia; y una API, que soportará la entrega segura de información a varios usuarios. Esta API incluye dos módulos, uno de transmisión y otro de recepción como se indica en la figura 3.1.

3.2. Señalización

Para llevar a cabo el módulo anteriormente descrito se han establecido una serie de requisitos, que son:

– Soportará el establecimiento de sesiones multimedia con otros equipos, para solicitar de éstos la recepción de información de telemetría.

– Gestionará las solicitudes de establecimiento de sesión recibidas desde otros equipos, e interactuará con el módulo de transmisión de su API local para configurar el reenvío de la información de telemetría a los equipos solicitantes.

– Estará basado en el protocolo estándar SIP, según se define en la RFC 3261 del IETF.

– El módulo de señalización identificará la información de telemetría mediante una Uniform Resource Identifier (URI) de SIP, que tendrá necesariamente el siguiente formato: *sip:Identificador-de-Usuario@Dominio-del-usuario:Puerto-del-dominio*

3.3. API

3.3.1. Módulo de Transmisión

El UAV, que usará la aplicación de emisor, enviará la información telemétrica al módulo de transmisión de la API, que será el encargado de enviar la información de telemetría con seguridad a los distintos receptores interesados de la sesión previamente establecida.

Este módulo debe ser capaz de enviar la información a varios receptores si éstos lo solicitan. La integridad de la información permanecerá intacta aunque el transmisor de los datos no sea la fuente original.

Requisitos del módulo de Transmisión

Para llevar a cabo el desarrollo del módulo de transmisión descrito anteriormente descrito es necesario cumplimentar los siguientes requisitos:

- La aplicación deberá encapsular los datos en paquetes y enviarlos a la API para poder transmitirlos.
- La comunicación entre la aplicación transmisora y la receptora deberá ser a través del protocolo UDP.

- La API implementará un método para enviar los paquetes usando protocolo UDP a los receptores. Este protocolo, UDP, es un mecanismo de comunicación no fiable.
- Este módulo deberá ser capaz de generar una seguridad a nivel de datos que permita identificar el origen de la información transmitida.
- Esta seguridad a nivel de datos es una firma digital de los paquetes encapsulados, que deberá ser enviada a todos los receptores involucrados en la conversación.
- Este módulo deberá asignar un número identificativo a los paquetes enviados para el futuro manejo de errores.
- La API tendrá que soportar que varios receptores soliciten la información, enviándoles a cada uno de ellos los paquetes y las firmas digitales generadas originales.
- Cuando varios receptores soliciten la información, no es necesario que sea el generador de los datos el que los transmita, sino que uno de los receptores pueda actuar como retransmisor de los datos originales y las respectivas firmas.
- El receptor que actúe como retransmisor de los datos originales deberá crear tantos canales de comunicación como usuarios se incorporen a la conversación. Estos usuarios recibirán los datos correspondientes a partir del momento de su incorporación en la comunicación.

3.3.2. Módulo de Recepción

Este módulo del sistema tiene la función de recibir los paquetes enviados por el emisor y gestionarlos dependiendo de la carga de datos que tengan. Una vez que tanto los paquetes de datos como el paquete de firma hayan sido recibidos, la API deberá proceder a la verificación de los mismos.

Requisitos del módulo de Recepción

Para llevar a cabo este módulo se deben tener en cuenta los siguientes requisitos:

- Los paquetes serán recibidos por la API a través del protocolo UDP y entregados a la aplicación receptora donde se encontrará el usuario.
- El módulo deberá ser capaz de organizar los paquetes recibidos según el número de secuencia y almacenarlos en memoria.
- Una vez que se recibe la firma, se comprobará si los datos almacenados en memoria corresponden a la firma, y se procederá a la verificación de los mismos.
- La API deberá ser capaz de manejar cualquier incidencia en la transmisión, como pérdida o retraso de los paquetes.

Capítulo 4

Implementación

En este capítulo se dará la información detallada del procedimiento seguido en la implementación del diseño del sistema propuesto en el capítulo 3. Para desarrollar la API se ha elegido el lenguaje de programación JAVA, dada la facilidad de implementación de los canales y la seguridad necesaria para cumplir con los requisitos del sistema. El nombre de la API desarrollada en este proyecto es **SecureSocket**. Una visión global de la implementación realizada se puede ver en la figura 4.1.

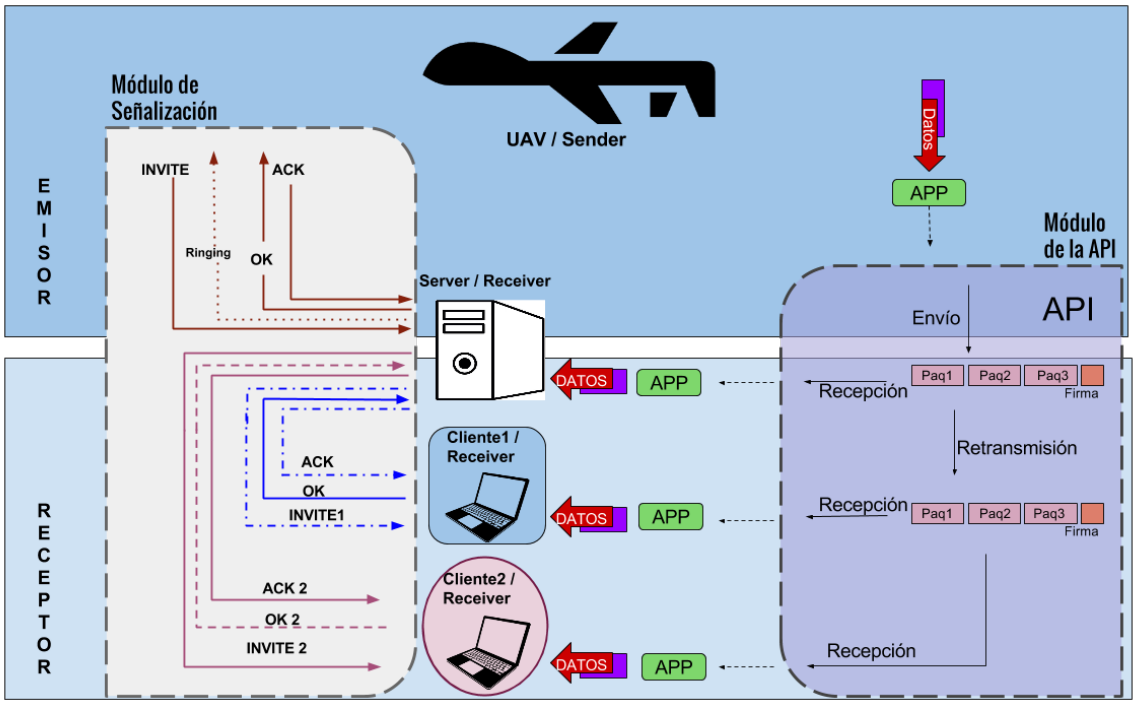

Figura 4.1: Implementación General del Problema

Tal y como se puede ver en la figura 4.1, la implementación constará de dos módulos: Señalización y API. Cada uno de estos módulos serán invocados por dos aplicaciones, una emisora y otra receptora de información. Se utilizará una comunicación entre ambos usuarios finales mediante UDP Socket, el cuál requiere un tipo de paquetes específico para encapsular los datos y enviarlos a través del Socket. Estos paquetes son denominados Datagram Packet.

4.1. Señalización

Para el desarrollo del módulo de señalización se han elegido los protocolos SIP y SDP, ya que permiten dar a conocer la intención de comenzar una comunicación entre un emisor y un receptor, además de negociar los parámetros del flujo multimedia que va a ser transmitido. El diagrama de flujo del desarrollo de este módulo está representado en la figura 4.2.

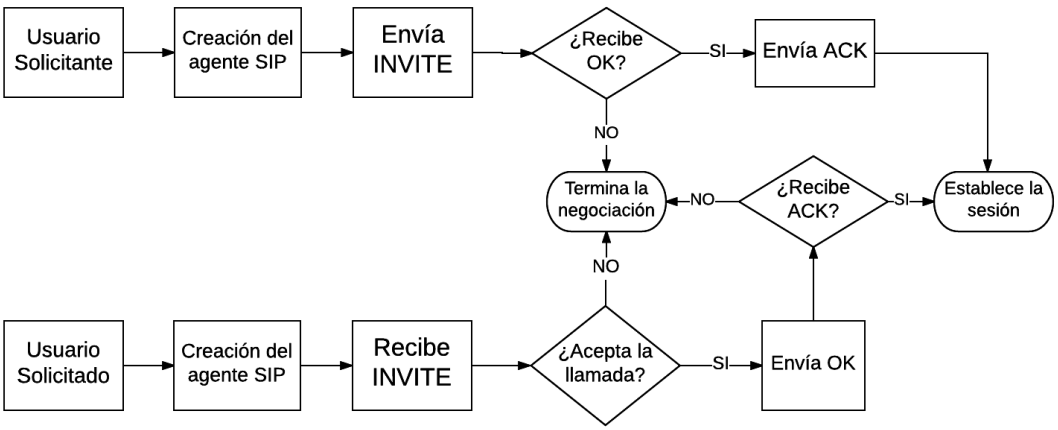

Figura 4.2: Diagrama de flujo: Señalización

Para este módulo se ha utilizado la API de JAVA: Jain SIP versión 1.2 [21]. Esta API es usada típicamente para el desarrollo de una aplicación en el lado del cliente (permite una comunicación entre clientes), ya que para desarrollar una aplicación en el lado del servidor se suele usar SIP Servlet API. Esta API contiene:

- Librerías:
  - *JainSipApi1.2.jar*: Donde se encuentras las interfaces y las clases principales.
  - *JainSipRi1.2.jar*: Referencia a la implementación SIP.
  - *log4j-1.2.8.jar*: Servicio de registro de accesos.
  - *concurrent.jar*: Utilidades de concurrencia.

- La interfaz *MessageProcessor*, que tiene los siguientes métodos:
  - **processMessage**: Recibe el nombre del emisor y el mensaje a enviar (INVITE, RINGING, OK, ACK, BYE).
  - **processError**: Maneja el mensaje de error ocurrido en la sesión.
  - **processInfo**: Recibe el mensaje de información sobre la sesión.
- La clase *SipLayer* contiene el constructor y métodos para desarrollar el módulo de señalización, y utiliza la interfaz *SipListener* de la librería *JainSipApi1.2.jar*.
  - La interfaz *SipListener* contiene los siguientes métodos:
    - **processRequest**: Procesa la solicitud de inicio de sesión.
    - **processResponse**: Procesa la respuesta a la solicitud.
    - **processTimeout**: Si en un tiempo no ha recibido respuesta, termina la solicitud de inicio de sesión.
    - **processIOException**: Maneja los errores que puedan ser causados en el envío de la solicitud o respuesta.
    - **processTransactionTerminated**: Procesa la petición de fin de sesión. Este método es ejecutado automáticamente en el fin del diálogo.
    - **processDialogTerminated**: Pone fin a la sesión. Este método también es ejecutado automáticamente.
  - La clase *SipLayer* contiene los siguientes métodos:
    - **SipLayer**: Es el constructor de la clase. Recibe el nombre, la dirección y el puerto del usuario a utilizar por el protocolo SIP. Con esta clase se crean una serie de propiedades que van a ser usadas para formar la cabecera de los mensajes informando al receptor quién es el solicitante. Además, crea un punto de escucha a las solicitudes SIP en el puerto recibido previamente. Por último, define que el protocolo a usar para el envío de las peticiones SIP es UDP.
    - **sendMessage**: Tiene como función crear los elementos principales, así como la creación del mensaje y el envío del mensaje. Recibe el nombre del usuario solicitado para el inicio de sesión, y el mensaje de solicitud. Entre los elementos principales se encuentran:
      - **Request URI**: Crea un identificador tanto para el solicitante como para el solicitado. Esta *sipURI* contiene el nombre, dirección y puerto del usuario.
      - **Cabecera de identificación de llamada**: Generea un identificador único de llamada.
      - **Cabecera de número de secuencia**: Añade un número de secuencia a la solicitud.
      - **Cabecera de origen**: Define quién es el originador de la petición.
      - **Cabecera via**: Informa de los nodos por los que pasa la petición.
      - **Cabecera max-forwards**: Define el número máximo de nodos por los que puede pasar antes de ser eliminada la petición.

- **processResponse**: Invocación al método de la interfaz *SipListener*. Este método es invocado por la pila de SIP cuando llega un mensaje de respuesta SIP. Recibe un evento llamado *ResponseEvent* que encapsula un objeto *Response*. En este método se comprueba si la respuesta del mensaje representa un éxito (con código 2xx) o representa un error. Luego se envía esta información de vuelta al usuario. Todas las respuestas usarán estos códigos, excepto la respuesta al INVITE que utilizará una respuesta especial, ACK.
- **processRequest**: Invocación al método de la interfaz *SipListener*. Este método también es invocado automáticamente por la pila de SIP, y recibe un objeto *RequestObject* el cual encapsula un objeto *Request*. Este método se encarga de analizar la petición, y luego genera y envía de vuelta una respuesta apropiada.
- Manejo de condiciones de error, son invocados automáticamente por la pila de SIP con una solicitud no puede ser enviada por razones específicas. Entre estos métodos se encuentran: **processTimeout**, invocado cuando la respuesta no llega a tiempo, y **processIOException**, invocado cuando existen errores de entrada y salida.

La API Jain SIP ha sido incluida en este proyecto de la siguiente manera:

- En las aplicaciones del **Emisor** y **Receptor**: Se invocan al método SIP creado en la API **SecureSocket**, pasándole como argumento el nombre, dirección y puerto del usuario.
- En la API **SecureSocket**:
  - Crea un nuevo usuario SIP para el **emisor** y para el **receptor** con los parámetros recibidos de las aplicaciones de los mismos, llamando al constructor de *SipLayer* comentado anteriormente. Con esta invocación, se construyen las cabeceras y las propiedades para el usuario específico.
  - Invoca al método modificado de la API el cual envía un mensaje de solicitud INVITE a la dirección de destino con la que se quiere comenzar una conversación.
- En la API Jain SIP:
  - No se ha modificado el constructor.
  - Se ha modificado el método *sendMessage* descrito anteriormente por el método *sendINVITE*, el cual fuerza a generar una solicitud INVITE, con los parámetros correspondientes del INVITE.
  - El método de *processRequest* no ha sido modificado, respondiendo a la solicitud del INVITE con un RINGING (código 180) y OK (código 200).
  - Se ha añadido un método que genera un ACK con sus parámetros correspondientes.

Este módulo, **Señalización**, interactúa con el módulo de la **API SecureSocket** para configurar los destinatarios en el *Módulo de Transmisión*.

4.2. API

4.2.1. Módulo de Transmisión

Esta sección describe los pasos tomados en la implementación para cumplir con los requisitos descritos en la sección 3.3.1. El diagrama de estados de este proceso está representado en la figura 4.3.

La aplicación del **emisor** se encarga de encapsular los datos a enviar en paquetes de tamaño fijo tipo Datagram Packet. Esta aplicación invoca al método **send** de la API **SecureSocket**, pasando por argumento los datagrampackets de los datos generados.

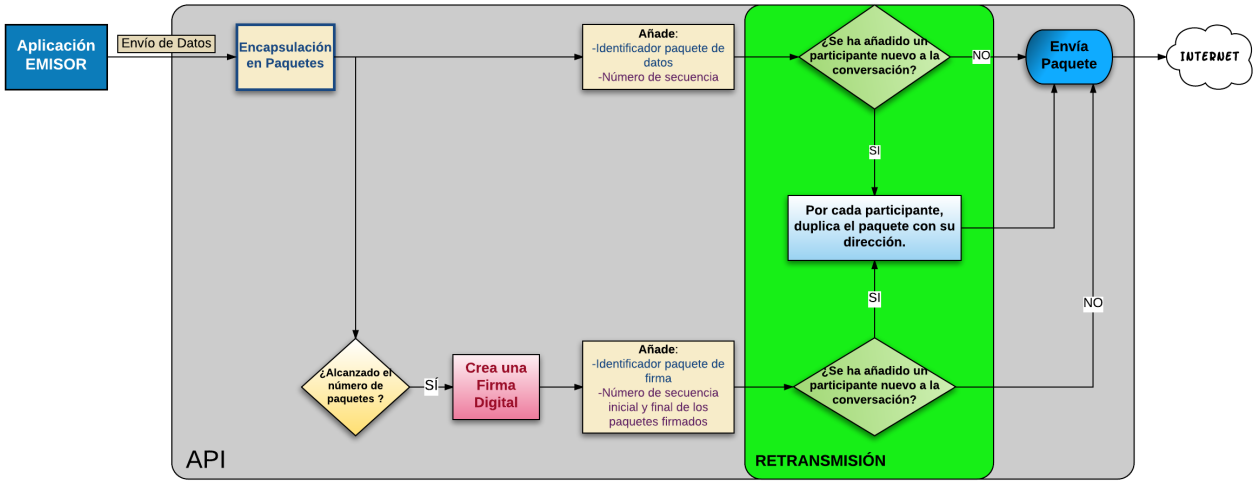

Figura 4.3: Diagrama Flujo: Módulo de Transmisión

Este módulo al ser invocado por primera vez, crea un *DatagramSocket* en un puerto específico por el que se enviarán los paquetes de datos. Además, genera las claves privada y pública necesarias para la generación y verificación de la firma. Estas claves son generadas con el método *generateKeys* que será explicado más adelante.

Para cumplir con los requisitos de gestión de retardos de recepción y pérdidas de paquetes, al comienzo de cada transmisión se genera un número aleatorio, denominado número de secuencia, que será incluido en los paquetes de datos, y servirán para la verificación de la firma.

Otros de los requisitos es implementar un método que haga posible la retransmisión de los datos originales, para ello se ha creado el método *addParticipant*. Aunque este método pertenezca al módulo de transmisión, es usado por los receptores de la información.

El módulo de transmisión está compuesto por los siguientes métodos:

- **send**:
  - Recibe el datagrampacket de la aplicación del *emisor*.
  - Invoca al método **datagrampacketextended** que se explicará más adelante. Este método modifica el paquete original.
  - Envía los paquetes modificados a través del *DatagramSocket* creado anteriormente.
  - Los datos pertenecientes a los paquetes enviados son almacenados en memoria en forma de vector.
  - Una vez que alcanza el número de paquetes establecido comienza el proceso de la generación de la firma de los datos.
  - Los datos guardados en el vector son pasados como argumentos al método **sign**, además de la clave privada generada al principio.
  - La devolución del método **sign** son los datos actuales de la firma, que es pasada como argumento al método **datagrampacketextended**, que encapsulará la firma en un Datagram Packet.
  - La firma encapsulada es enviada a través del *DatagramSocket* creado anteriormente.
- **datagrampacketextended**:
  - Tiene como argumentos: *DatagramPacket*, *identificador de paquete*, *número de secuencia mínimo* y *número de secuencia máximo*.
  - El *identificador de paquete* indica si el paquete que se modifica es datos(0) o de firma (1), este identificador, de longitud 1 byte, se añadirá al comienzo de los datos encapsulados en el Datagram Packet.
  - Si pertenece a un paquete de datos, en la segunda posición de los datos se añadirá el número de secuencia correspondiente a ese paquete.
  - Si pertenece a un paquete de firma, en la segunda y tercera posición se añadirá los números de secuencia (*número de secuencia mínimo* y *número de secuencia máximo*), correspondientes al primer y último paquete de datos utilizados para generar la firma.
- **sign**:
  - Recibe por argumento un array de bytes y la clave privada del receptor.
  - El array de bytes pertenece a los datos que contenían los Datagram Packet.
  - Mediante el proceso de generación de firma que se explicará a continuación, se firman los datos y se les aplica una función hash.
- **addParticipant**:
  - Recibe por parámetro la dirección IP de los participantes que solicitan unirse a la conversación.
  - Almacena en memoria las direcciones de los diferentes participantes que desean unirse, que serán usadas en el módulo de retransmisión para añadírselas a los paquetes.

4.2.2. Generación de la firma

Este apartado contiene cómo se han creado los métodos **generateKeys** y **sign** mencionados en la sección 4.2.1, haciendo uso de la API de JAVA, incluida en el Java Development Key (JDK) versión 1.8, importada como paquete llamado *java.security*, que permite el uso de funciones y algoritmos específicos de seguridad.

Para el método que se ejecuta al principio de la transmisión, **generateKeys** se utiliza *KeyPairGenerator* procedenete de la API de seguridad, lo que permite elegir el tipo de algoritmo para la generación de las claves. En nuestro proyecto se ha usado el algoritmo RSA. Una vez seleccionado el algoritmo, se invoca al método *generateKeyPair()* de la clase *KeyPairGenerator*.

En nuestro proyecto se ha optado por la generación de las claves con una semilla compartida por el emisor y receptor, lo que genera siempre el mismo par de claves. Con este procedimiento no es necesario la transmisión de la clave pública del emisor al receptor, provocando una mayor simplicidad en el apartado de pruebas. Si el receptor no sabe la clave pública del emisor, este último tiene que mandarla a través del Socket UDP y el receptor almacenarla para cuando necesite hacer uso de ella en la verificación de las firmas. Si el emisor no quiere que nadie fuera de la conversación sepa su clave pública, deberá hacer uso de algún algoritmo de cifrado propuesto en el apartado 2.

Después de la generación de las claves, el método **send** ya es capaz de generar firmas. Para ello se ha desarrollado el método **sign** que recibirá por parámetros los datos y la clave privada de la fuente de la información. Este método utiliza la clase *Signature* de la API de seguridad del JDK. Para la generación de la firma se ha elegido el algoritmo RSA y el algoritmo hash SHA1, aunque es posible utilizar otros algoritmos, tanto de hash como de firma, soportados por la API de seguridad. Entre los algoritmos que pueden ser usados se encuentran:

Cuadro 4.1: Algoritmos soportados

RSA	**DSA**
MD2withRSA	SHA1withDSA
MD5withRSA	SHA1withECDSA
SHA1withRSA	SHA256withECDSA
SHA256withRSA	SHA384withECDSA
SHA384withRSA	SHA512withECDSA
SHA512withRSA	

4.2.3. Módulo de Recepción

Mediante el Socket UDP se reciben todos los paquetes enviados por el emisor. Al llegar éstos, se comprueba si pertenecen a un paquete de datos o de firma. Si forman parte del grupo de datos, se van almacenando a la espera de la firma. Una vez que el Datagram Packet que contiene la firma ha sido recibido, se procede a la comprobación de la misma. (Este proceso será explicado en el siguiente apartado 4.2.4). El diagrama de estados correspondiente al *Módulo de Recepción* está representado en la figura 4.4

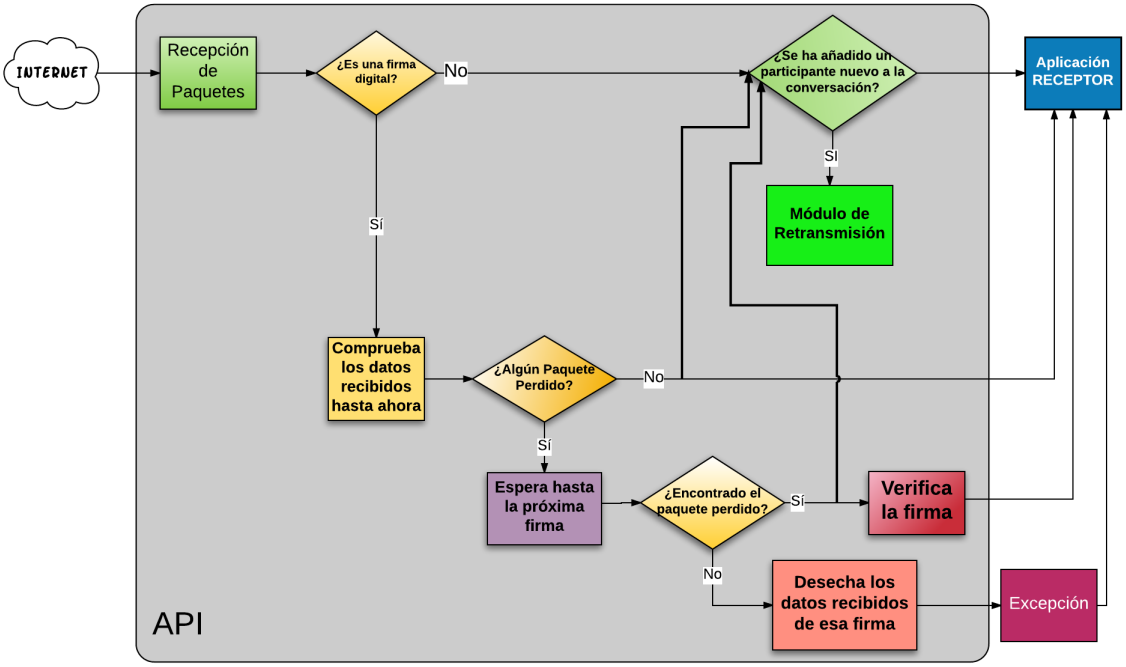

Figura 4.4: Diagrama Flujo: Módulo Receptor

Este módulo implementa los siguientes métodos:

- **receive**
  - Recibe a través del Socket UDP los Datagram Packet que contienen la información encapsulada.
  - Comprueban el primer byte de los datos del paquete, si el byte es 0 pertenece a un paquete de datos y lo almacena en memoria; si el byte es 1 pertenece a un paquete de firma.
  - Si el paquete corresponde a una firma, procede a comprobar que ha recibido todos los paquetes de datos correspondientes a esa firma. Este procedimiento se efectúa con el método *checkPackets*.
  - Si la comprobación de los paquetes de datos y la firma son correctos, se procede a la verificación de los mismos con el método *verify*.

- **checkPackets**
  - Recibe los datos almacenados y la firma.
  - Mediante el número de secuencia introducido en el *Módulo de Transmisión* se comprueba que el número de secuencia del primer paquete corresponde con el primer número almacenado de la firma, y se van comprobando que los demás números de secuencia incrementan en 1 hasta alcanzar el segundo número almacenado de la firma.
  - Si se detecta alguna irregularidad en la comprobación del número de secuencia, se advierte al método *receive* que falta el paquete con ese número de secuencia faltante.
  - El método *receive* comprobará todos los paquetes que reciba a partir de entonces con el método *isMyPacket*.
  - Los datos recibidos son almacenados en una memoria temporal a la espera del paquete perdido.
- **isMyPacket**
  - Una vez que el método *send* le da la orden de comprobar el número de secuencia del paquete perdido, comprueba todos los paquetes que son recibidos con el número de secuencia.
  - Si encuentra el paquete perdido, lo almacena juntos los otros datos pertenecientes a la firma para ser verificados.
  - Si recibe un paquete de firma antes de recibir el paquete con el número de secuencia perdido, descarta los datos almacenados en la memoria temporal.
- **verify**
  - Recibe por argumento el texto firmado, los datos recibidos y la clave pública.
  - Se emplea la API de seguridad que se explica a continuación para la verificación del texto firmado con la clave pública, comparándolo con el texto plano (pertenecientes a los datos recibidos).

Un usuario externo a la comunicación puede solicitar la recepción de la información a otro receptor. Este usuario externo, hace la petición mediante el *Módulo de Señalización*. Una vez que se ha aceptado la petición, el receptor invoca al método **addParticipant** con la dirección IP recibida del *Módulo de Señalización*. Con el método **addParticipant** se le informa al método **receive** de la API, que tiene que dar los paquetes recibidos al *Módulo Transmisor*, para que éste duplique los paquetes con la dirección IP correspondiente a los nuevos incorporados.

4.2.4. Verificación de la firma

Este proceso implementa también la API de seguridad del apartado 4.2.2. En este caso utiliza la parte de verificación de la firma.

El receptor, en posesión de la clave pública de la fuente de la información, puede verificar los datos recibidos con la firma. Esta verificación se hace a nivel de la API **SecureSocket**. El receptor sólo es consciente de si la firma es correcta o no mediante el uso de excepciones, lanzadas a la aplicación usada por el receptor. Si hay un error en la comprobación de la firma, se lanza una excepción a la aplicación informando al usuario que ha habido un fallo en la verificación de la firma y que esos datos no están disponibles.

Para esta sección se ha utilizado la clase **Signature** del paquete de seguridad, con la cual elegimos usar el algoritmo empleado para la generación de la firma: *SHA1withRSA*. Una vez que tenemos el algoritmo elegido, se procede al inicio de la verificación con la clave pública de la fuente de la información, comparando el texto firmado con el texto plano de los datos recibidos.

Capítulo 5

Validación y Pruebas

En este capítulo se describen los diferentes tipos de pruebas realizadas para verificar el correcto funcionamiento de la API implementada, así como los resultados obtenidos.

5.1. Entorno de pruebas

El entorno en el que se han desarrollado las pruebas, ha sido en el laboratorio del Departamento de Telemática de la Universidad Carlos III de Madrid, el cual está compuesto por equipos con las siguientes características:

- Procesador i3-3240.
- Tamaño Random Access Memory (RAM) 4GB.
- 32 bits.
- Tarjeta de red 1Gb/s.

5.2. Pruebas de Rendimiento

Estas pruebas se han realizado en el entorno descrito en el apartado 5.1, entre dos equipos, uno actuando como fuente y otro como receptor, utilizando los procesos de envío y recepción de la API respectivamente.

5.2.1. Firma por cada paquete de datos

En esta prueba se ha procedido a generar una firma por cada paquete enviado. Aquí se puede apreciar la potencia del equipo en la generación de la firma, esto es, siguiendo la figura 5.1 existen unos picos de bajada de velocidad justo antes de ser enviado el paquete de firma. Así mismo, es posible apreciar en la gráfica de recepción la eficiencia de la velocidad de estos paquetes obteniendo una velocidad de 2,47 Mbps en la recepción.

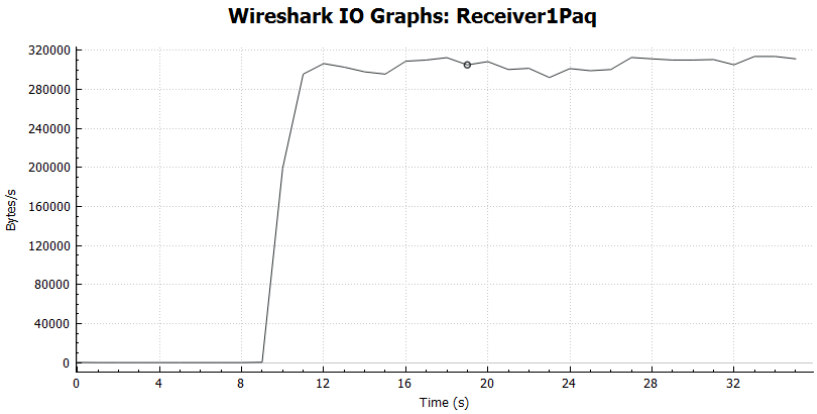

Figura 5.1: Recepción Con Firma Cada Paquete

El cálculo del promedio del tiempo existente entre paquetes recibidos es de 1'99 milisegundos con una desviación típica de $2x10^{-3}$.

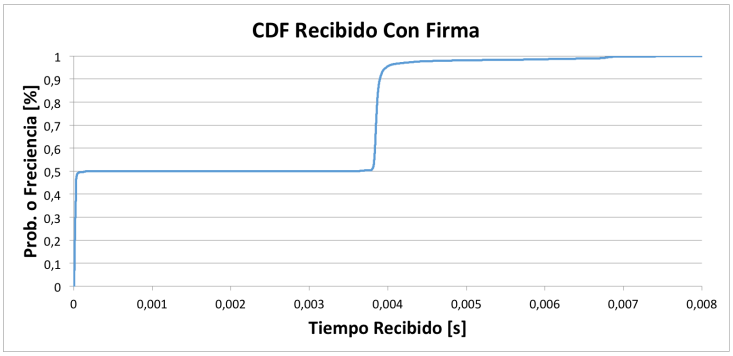

Figura 5.2: CDF: Recepción Con Firma Cada Paquete

Con la gráfica de la figura 5.2 se puede apreciar que el 50 % de las veces la transmisión tarda más de 4 milisegundos. Estos datos son debidos a que el tiempo que tarda en generar una firma son de 4 ms y el 50 % de los paquetes enviados corresponden a una firma.

5.2.2. Firma por cada dos paquetes de datos

En esta prueba se ha ampliado a dos paquetes de datos por firma. La velocidad de recepción es de 4,5Mbps, superior al caso anterior. Este proceso tiene menor coste computacional para el equipo, lo que implica un aumento en la velocidad de transmisión. La generación de la firma se puede apreciar en la gráfica en los picos de bajada de velocidad, tiempo que toma el equipo en generar la firma.

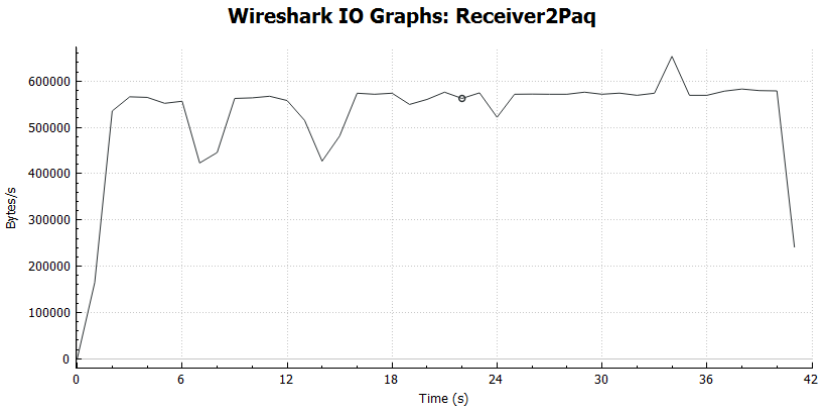

Figura 5.3: Recepción Con Firma Cada Dos Paquetes

El promedio de tiempo de recepción cuando se genera una firma cada dos paquetes es de 1,36 milisegundos, tiempo menor que el tomado en la sección 5.2.1, con una desviación típica de $1,9x10^{-3}$.

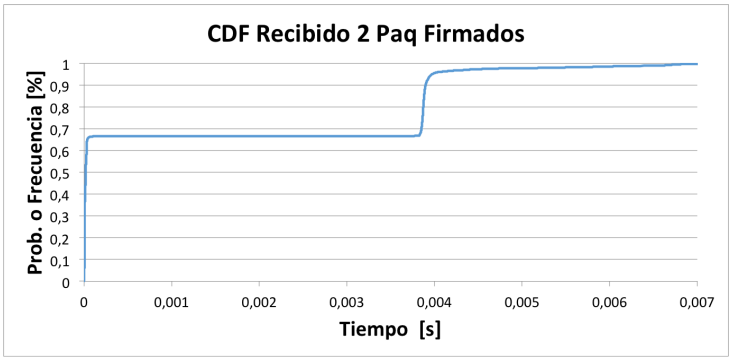

Figura 5.4: CDF: Envío y Recepción Con Firma Cada Dos Paquetes

En la figura 5.4 es posible ver cómo va creciendo la probabilidad (hasta casi el 70 %) de que el tiempo de transmisión sea menor a 4 milisegundos. Esto es debido a que 1 de cada 3 paquetes enviados pertenece a una firma.

5.2.3. Firma por cada cuatro paquetes de datos

Para la realización de esta prueba se ha generado un paquete de firma por cada cuatro paquetes de datos enviados. En esta prueba el rendimiento de la API es de 8,77 Mbps para la recepción, notándose que el coste computacional para el equipo disminuye.

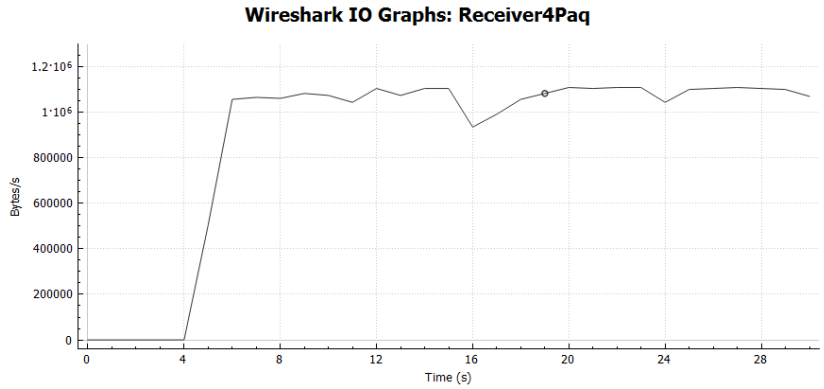

Figura 5.5: Envío y Recepción Con Firma Cada Cuatro Paquetes

El promedio de tiempo que tarda en recibir los paquetes es de 0,8 milisegundos con una desviación típica de $1,6x10^{-3}$.

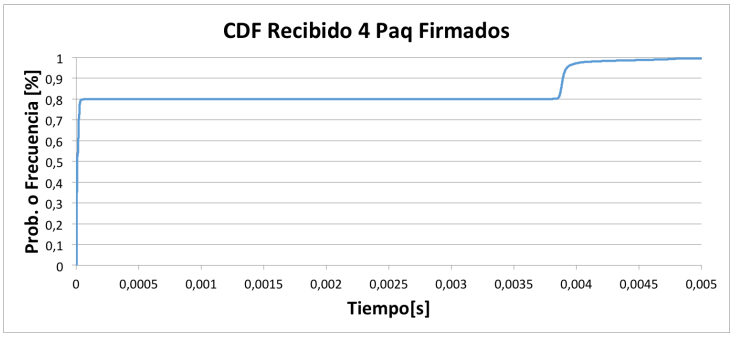

Figura 5.6: CDF: Recepción Con Firma Cada Cuatro Paquetes

Según se puede comprobar en la figura 5.6 se puede ver que la probabilidad ha aumentado hasta el 80 % de que el tiempo sea menor de 4 milisegundos, debido a que 3 de 4 paquetes son paquetes de datos.

5.2.4. Firma por cada seis paquetes de datos

El proceso de firma por cada seis paquetes se ha generado para comprobar el rendimiento del sistema vista la evolución de la velocidad respecto a los procesos anteriores. En este caso la velocidad de recepción es de 12,83 Mbps, siendo unos valores muy positivos para el desarrollo de la aplicación.

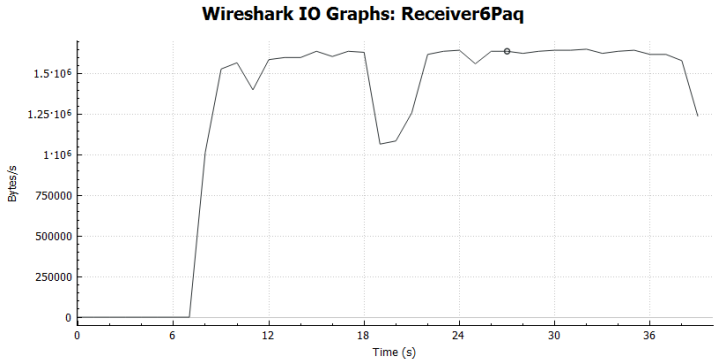

Figura 5.7: Recepción Con Firma Cada Seis Paquetes

La media del tiempo de recepción en este proceso es de 0,589 milisegundos con una desviación típica de $1,45x10^{-3}$. Haciéndose notable la reducción con los procesos de las pruebas anteriores.

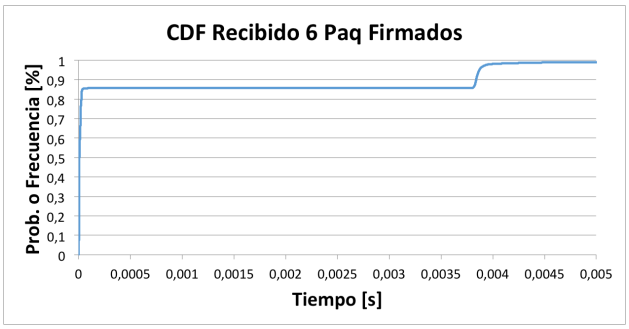

Figura 5.8: CDF: Recepción Con Firma Cada Seis Paquetes

Puesto que 1 de cada 6 paquetes pertenece a la firma, en la gráfica 5.8 se puede percibir el aumento de la probabilidad hasta rozar el 90 %, que el tiempo de transmisión sea menor a 4 milisegundos.

5.2.5. Firma por cada ocho paquetes de datos

El rendimiento de la velocidad de transmisión del proceso en el cual se genera una firma por cada ocho paquetes recibidos es de 16,83 Mbps. Este proceso, generar una firma por cada ocho paquetes, supone un coste computacional menor que los procesos anteriores reflejándose en la velocidad de transmisión.

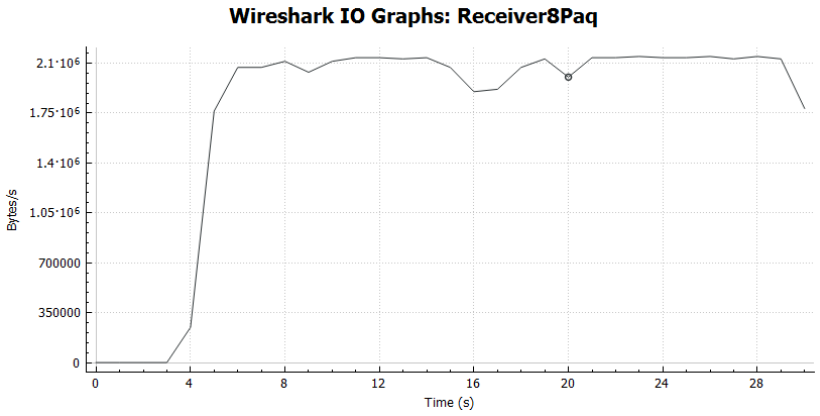

Figura 5.9: Recepción Con Firma Cada Ocho Paquetes

El tiempo medio de transmisión es de 0,45 milisegundos con una desviación típica de $1,2x10^{-3}$.

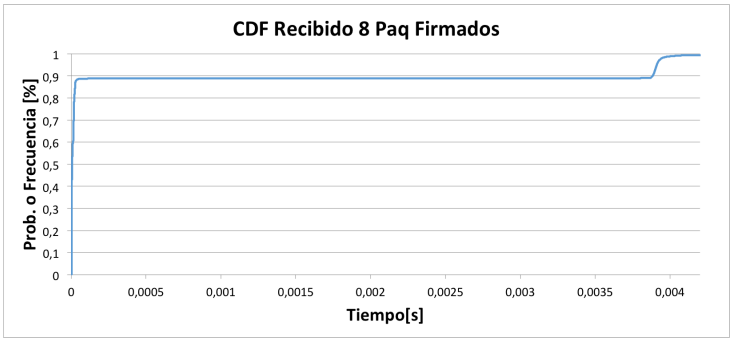

Figura 5.10: CDF: Recepción Con Firma Cada Ocho Paquetes

5.2.6. Firma por cada dieciséis paquetes de datos

La siguiente prueba no es recomendable para el envío de datos en tiempo real, ya que generar un paquete de firma por cada dieciséis paquetes de datos puede conllevar a no poder verificar la firma dado que puede haber pérdidas de los paquetes enviados. Aún así, se ha procedido a realizar esta prueba para ver el rendimiento del equipo cuando genera una firma por cada 16 paquetes.

En esta prueba se ha obtenido una velocidad de transmisión de 32,26 Mbps, siendo un valor mucho mayor que el obtenido en la prueba 5.2.1, pero aún sigue siendo mucho menor que el obtenido en la prueba **??**.

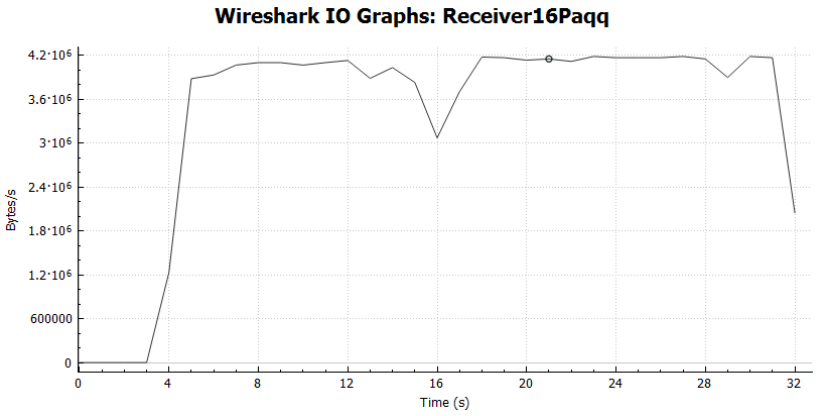

Figura 5.11: Transmisión Con Firma Cada Dieciséis Paquetes

El promedio de tiempo de recepción en la prueba de generación de paquete de firma por cada dieciséis paquetes de datos es de 0,246 milisegundos con una desviación típica de $9,74x10^{-3}$.

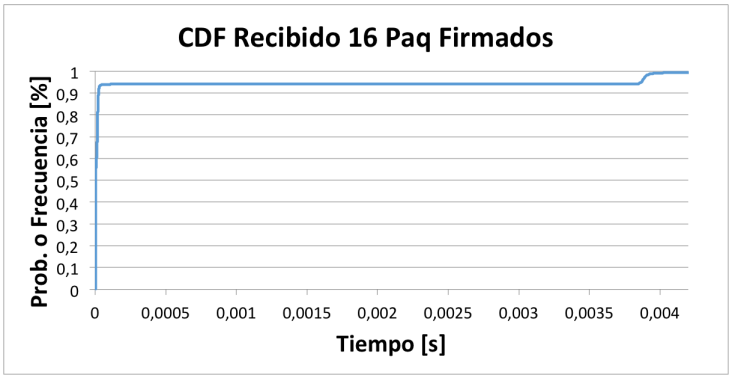

Figura 5.12: CDF: Recepción Con Firma Cada Dieciséis Paquetes

En la figura 5.12 se puede observar cómo la probabilidad de que el tiempo de envío sea mínimo, alcanzando valores superiores al 90 %.

5.2.7. Tabla Comparativa de Resultados

En esta sección se muestra una tabla comparativa con los resultados obtenidos de las pruebas.

Los valores obtenidos de las pruebas anteriores no indican el throughput de la información, sino de la totalidad de los paquetes recibidos, esto significa que se está obteniendo la velocidad de transmisión teniendo en cuenta las firmas. Los paquetes de datos generados

por la API son de 1000 octetos, mientras que los de firma son de 130 octetos. Al usar el protocolo UDP para la transmisión de los paquetes, éste añade unas cabeceras a los mismos incrementando su tamaño hasta los 1042 octetos para los paquetes de datos y 172 octetos para los de firma. Por lo tanto, la *tasa efectiva* corresponde a la velocidad por la eficiencia de los paquetes de datos:

$$\eta_e = \frac{e_0}{e_f}$$

Dónde η_e es la tasa efectiva, e_0 es el tamaño del paquete original, y e_f es el tamaño final.

Pruebas	Velocidad	Tiempo Medio	Tasa Efectiva
Prueba I: Firma 1 Paquete	2,47 Mbps	2 ms	2,03 Mbps
Prueba II: Firma 2 Paquetes	4,5 Mbps	1,36 ms	3,71 Mbps
Prueba III: Firma 4 Paquetes	8,77 Mbps	0,8 ms	27,22 Mbps
Prueba IV: Firma 6 Paquetes	12,83 Mbps	0,59 ms	10,56 Mbps
Prueba V: Firma 8 Paquetes	16,83 Mbps	0,45 ms	13,86 Mbps
Prueba VI: Firma 16 Paquetes	32,26 Mbps	0,246 ms	26,57 Mbps

Cuadro 5.1: Tabla Comparativa.

5.3. Retransmisión de Datos

Para esta prueba se ha dispuesto de tres ordenadores tal y como presenta la figura **??**. Uno de los ordenadores ha actuado de fuente de información, el equipo del medio ha recibido y verificado la información a la vez que la retransmitía al tercer equipo, y este último equipo recibía la información generada por la fuente y verificaba que los datos eran originarios de la fuente.

Para esta prueba se ha utilizado el programa *Wireshark*, obteniendo la siguiente captura:

```
23 0.020134  163.117.144.202  163.117.144.206  UDP   173 30000 → 20000  Len=131
24 0.020159  163.117.144.202  163.117.144.206  UDP  1043 30000 → 20000  Len=1001
25 0.020169  163.117.144.206  163.117.144.207  UDP   173 20000 → 50000  Len=131
26 0.020417  163.117.144.206  163.117.144.207  UDP  1043 20000 → 50000  Len=1001
27 0.024007  163.117.144.202  163.117.144.206  UDP   173 30000 → 20000  Len=131
28 0.024032  163.117.144.202  163.117.144.206  UDP  1043 30000 → 20000  Len=1001
29 0.024048  163.117.144.206  163.117.144.207  UDP   173 20000 → 50000  Len=131
30 0.024315  163.117.144.206  163.117.144.207  UDP  1043 20000 → 50000  Len=1001
```

Figura 5.13: Retransmisión de un usuario receptor

Como se puede comprobar en la figura 5.13, los paquetes coloreados en rosa son enviados por la fuente (163.117.144.202) y recibidos por el equipo retransmisor (163.117.144.206), mientras que los paquetes en amarillo son los retransmitidos por el equipo retransmisor (163.117.144.206) y recibidos por el equipo receptor final (163.117.144.207). Tal y como se puede ver en la captura, se está generando un paquete de firma por cada paquete enviado.

5.4. Gestión de retardos de recepción y pérdidas de paquetes

Para esta prueba se han intercambiado el número de secuencia de dos paquetes antes de ser enviados, los cuales pertenecen a firmas distintas.

- **Prueba I** – El paquete llega antes que la siguiente firma: La aplicación espera al paquete perteneciente a la firma correspondiente. Cuando dicho paquete llega, es posicionado en el lugar correspondiente para la comprobación de la firma. Dicho proceso se representa en la figura 5.14.

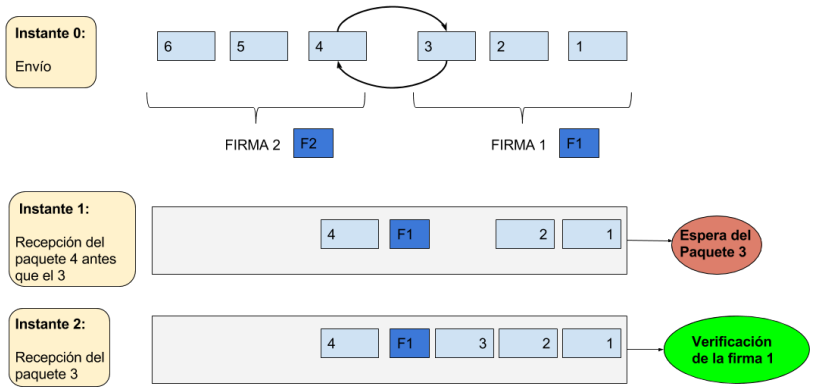

Figura 5.14: Gestión en el retraso de paquetes

- **Prueba II** – El paquete llega después que la siguiente firma: En este caso, cuando la aplicación recibe la firma correspondiente al próximo conjunto de datos, elimina de la memoria los datos anteriores, y procede a verificar la nueva firma. Dicho proceso se representa en la figura 5.15.

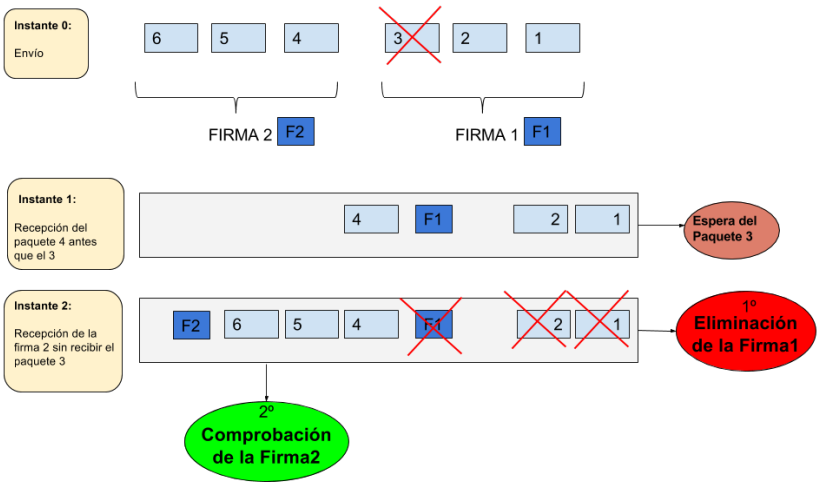

Figura 5.15: Gestión en la pérdida de paquetes

Ambas pruebas han resultado satisfactorias, cumpliéndose así los requisitos del diseño propuesto respecto a la pérdida y retraso de paquetes.

5.5. Establecimiento de Sesión

Esta prueba consiste en verificar el correcto funcionamiento del establecimiento de la sesión. Para ello se ha requerido el uso del programa *Wireshark*, el cual es capaz de monitorizar y capturar el tráfico de la red, filtrando los paquetes de SIP. Las siguientes capturas corresponden con el diálogo de SIP que establece un equipo para solicitar la información de telemetría a otro equipo, tal y como se muestra en la figura 5.16:

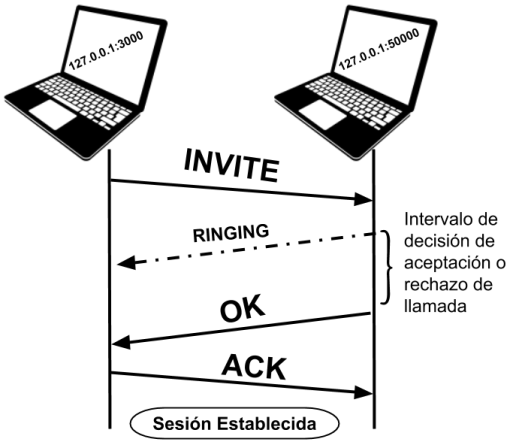

Figura 5.16: Establecimiento de la Sesión

- **I – INVITE:**

Figura 5.17: Envío de Solicitud de Comunicación

- **II – RINGING:**

Figura 5.18: Señalización llegada de la Solicitud

- **III – OK:**

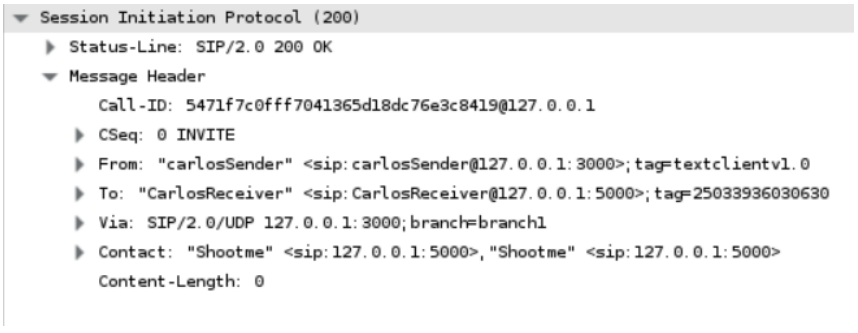

Figura 5.19: Respuesta Positiva a la Solicitud

- **IV – ACK:**

Figura 5.20: Establecimiento de la Comunicación

Capítulo 6

Gestión del proyecto

Este apartado recoge los aspectos más relevantes relacionados con el proyecto, tales como la planificación; un desglose de las distintas tareas realizadas a lo largo de este trabajo fin de grado, y el coste asociado al proyecto.

6.1. Planificación

Debido a la extensión del Trabajo Fin de Grado, se optó por dividir el proyecto en cinco fases principales. Las fases con sus respectivas tareas y las horas empleadas, están representadas en el cuadro 6.1. Las fases principales son:

1. Estudio Previo.

2. Diseño API.

3. Desarrollo e implementación.

4. Pruebas y documentación.

5. Realización de la memoria

Las distintas fases se han desarrollado de manera no lineal, como se puede observar en el diagrama de Gantt, Figura 6.1. Es fácilmente observable el solapamiento de algunas tareas, debido a cambios durante el diseño así como en la implementación. La redacción de la memoria no se comenzó hasta que todas las pruebas fueron concluidas.

Fases y Tareas	Horas empleadas
1. Estudio Previo	
I. UAV	15 h
II. Seguridad: Codificación y Firmas Digitales	30 h
III. Protocolos: UDP, RTP, SIP y SDP	15 h
IV. ICN	15 h
2. Diseño API	
I. Requisitos API :	
I.i Modulo emisor	5 h
I.ii Modulo Receptor	5 h
I.iii Retraso y Pérdida de paquetes	15 h
II. Establecimiento de conexión	15 h
3. Desarrollo e implementación	
I. Modulo emisor	30 h
II. Modulo Receptor	45 h
III. Retraso y Pérdida de paquetes	40 h
IV. Retraso y Pérdida de paquetes	30 h
4. Pruebas y su documentación	
I. Rendimiento:	
I.ii con firma:	
a. 1 paquete	5 h
b. 2 paquetes	5 h
c. 4 paquetes	5 h
d. 6 paquetes	5 h
e. 8 paquetes	5 h
f. 16 paquetes	5 h
I.iii Comparación	10 h
III. Establecimiento de conexión	10 h
IV. Retraso y Pérdida de paquete	10 h
5. Realización de la Memoria	
I. Redacción	70 h
II. Corrección y Maquetación	20 h
Total Horas	410 h

Cuadro 6.1: Fases y tareas del proyecto.

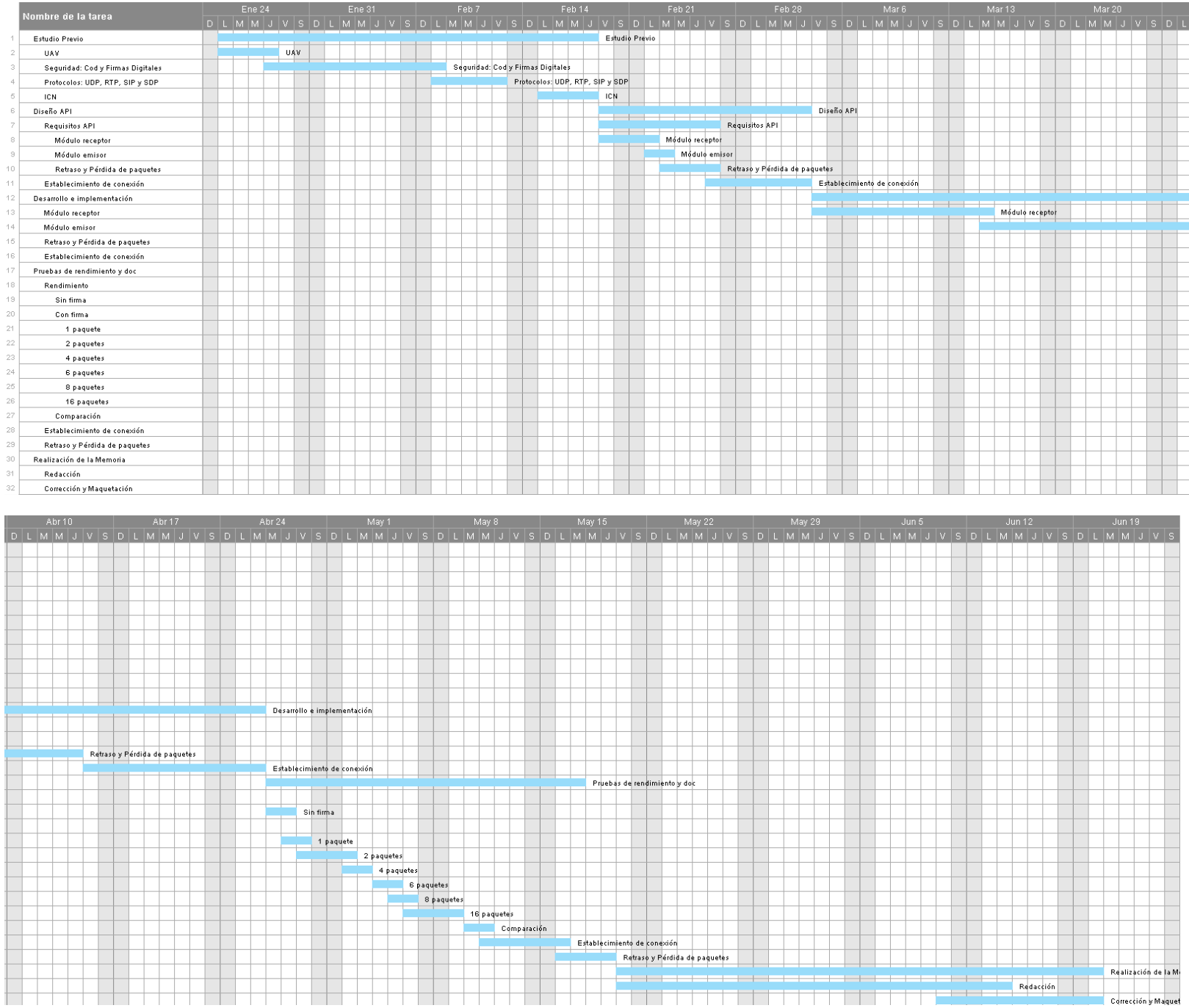

Figura 6.1: Diagrama de Gantt del proyecto.

6.2. Análisis económico

Costes Materiales

En estos costes están presentes tanto los dispositivos empleados para el desarrollo como los dispositivos utilizados para las pruebas. También se ha añadido el coste cables ethernet y otros componentes en el apartado "otros", y el coste de una conexión a Internet estable durante los meses de implementación y pruebas del proyecto.

Elementos	**Precio (€)**
Herramientas Software	
GNU/Linux	Software libre
Windows	135
Eclipse	Software libre
Wireshark	Software libre
Dispositivos Hardware	
Portátil Lenovo	750
Asus procesador i3-3240, 4GB ram y 32 bits	350 x2
Otros Materiales	35,95
Conexión a Internet 4 meses	95
Total	1.715,95

Cuadro 6.2: Precio de los elementos utilizados software y hardware durante el proyecto.

Costes de Personal

Para los cálculos de coste de personal se ha tenido en cuenta las horas trabajadas por el Ingeniero Junior a una media de 4 horas al día debido a que éste se encontraba en el último curso del grado, y en el caso del Ingeniero Senior se contabilizan las horas de tutorías, además de las horas empleadas en corrección de la memoria.

Nombre	**Personal**	**Horas**	**Precio/Hora (€/h)**	**Importe (€)**
Iván Vidal Fernández	Ingeniero Senior	50	33,00	1.815
Carlos M. Castillo Mateos	Ingeniero Junior	410	20,50	8.405
			Total	10.220

Cuadro 6.3: Coste de Personal.

Coste Total

En los costes totales del proyecto se encuentran tantos los costes de material y personal calculados anteriormente, además de los costes indirectos del 20 % entre los cuales se encuentran gastos de electricidad, agua y puestos del laboratorio de telemática de la Universidad Carlos III de Madrid.

Concepto	**Precio (€)**
Coste Material	1.715,95
Coste de Personal	10.220,00
Costes indirectos (20 %)	2.387,19
Subtotal	**14.323,14**
IVA (21 %)	3.007.86
Total	**17.331**

Cuadro 6.4: Coste Total del proyecto.

Capítulo 7

Conclusión y Líneas Futuras

En este capítulo se incluye una conclusión general del proyecto, así como los diferentes problemas encontrados en la elaboración, y la calidad de los resultados obtenidos. Así mismo, contiene las posibles mejoras y líneas futuras de trabajo.

7.1. Conclusión general

El principal objetivo de este Trabajo Fin de Grado ha consistido en el diseño y desarrollo de un sistema para la entrega segura de información en entornos de UAVs. Para ello, se ha realizado un diseño modular, consistente en un módulo de *señalización* y una **API**, que permite soportar la distribución segura de contenido a múltiples usuarios.

La API desarrollada en este proyecto puede ser utilizada en cualquier tipo de transmisión basada en el protocolo UDP; por ejemplo, video, información de sensores u otra información de telemetría. Sólo se necesita invocar el método requerido sin necesidad de previos conocimientos sobre seguridad. Esta API permite asegurar la procedencia y la fuente de la información, notificando al receptor si la fuente ha sido suplantada y rechazar los datos. Se puede utilizar en ambos extremos de la comunicación, tanto para la transmisión de la información como en la recepción.

Este proyecto se diseñó por módulos debido a las diferentes posibilidades de uso de la API. La primera parte a realizar fue la comunicación entre dos usuarios mediante el protocolo UDP, dónde el emisor puede elegir la longitud de los paquetes y el tiempo de transmisión entre paquetes.

En segundo lugar, se introdujo la seguridad. Después de pensarlo detenidamente, se decidió introducir en su propio paquete los datos generados tras la firma, y la utilización del algoritmo RSA (en la API se puede cambiar de algoritmo fácilmente). Después de esta decisión, se procedió a la creación del par de claves, priva y pública, por parte del emisor. La clave privada es utilizada para generar la firma, así como clave pública es utilizada para la verificación de la firma, y debe estar en posesión del receptor.

En tercer lugar, se realizó la parte de señalización dónde se utilizó el protocolo SIP. Gracias a esta señalización, un usuario es capaz de solicitar unirse a una conversación y recibir la información intercambiada en esa sesión.

Para poder cumplir con uno de los requisitos esenciales del proyecto, *retransmisión*, la API permite que cualquier usuario, dentro de la conversación, pueda actuar como emisor y reenviar los datos originador por la fuente a los todos los solicitantes.

Por último, como las pruebas se realizaron en un entorno ideal, sin posibilidad de tener errores, se forzó un entorno hostil y se diseñó una metodología para la gestión de retardos de recepción y pérdidas de paquete. Estas pruebas resultaron muy satisfactorias para el autor, ya que provocar un entorno hostil y desarrollar el proceso del manejo de errores resultó una tarea ardua.

Los resultados obtenidos en el apartado 5 de validación y pruebas, muestran que, a pesar de la penalización en prestaciones introducida por el proceso de firma de paquetes, las prestaciones obtenidas son apropiadas para la distribución de información de telemetría, incluso realizando una firma por paquete. A modo de ejemplo, la información de telemetría distribuida en el sistema UAV SIVA del INTA, requiere la transmisión de vídeo a 1 Mbps, mientras que la información de sensores se envía como telemetría en tasas del orden de Kbps. En caso de requerirse la transmisión a tasas mayores, la solución presentada permite asegurar el origen de la información aumentando la granularidad en el proceso de firma de paquetes.

7.2. Líneas Futuras de Trabajo

A continuación, se mencionan posibles mejoras y líneas de trabajo futuras para este proyecto:

- **Ejecución de pruebas sobre un entorno de UAVs real**. A este efecto, una línea de trabajo a realizar sería probar la ejecución del sistema desarrollado sobre la maqueta de pruebas de UAVs disponible en el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid, que debido a limitaciones de tiempo, estas pruebas no han podido ser realizadas.
- **Ampliación del sistema.** Extensión del sistema para soportar el descubrimiento de equipos que reciben la información de telemetría, y que por tanto pueden utilizarse para acceder a la misma en tiempo real. Con esta ampliación del sistema, se podría elegir entre los candidatos al equipo con el que contactar.
- **Soporte de nuevos protocolos de autenticación.** La versión actual del software se basa en el uso de los algoritmos RSA y SHA1 para la generación de firmas digitales. Futuras versiones podrían incorporar nuevos algoritmos a este respecto como por ejemplo AES como algoritmo de firma, y otras variantes de SHA, como SHA256.
- **Soporte de confidencialidad**. Actualmente el sistema permite autenticar el origen de la información recibida. Una futura línea de mejora consistiría en cifrar la información en origen para, de este modo, garantizar la confidencialidad de la misma y desarrollar un mecanismo para la distribución de la clave de cifrado a los receptores autorizados.

Apéndices

Apéndice A

Introduction

This chapter is the English translation of chapter 1. Here it is discussed the main aspects of the project, so as the goals, the scope and the motivation to fulfil it. Later, the document structure will be introduced.

A.1. Motivation of the project

An UAV is an Unmanned Aerial Vehicle that does not need in-flight crew, but it needs a ground crew in a Ground Control Station or GCS.

There exists many kinds of UAV depending on the goal of their design, with different ranges of action coverage, as different capabilities depending on their payload (infrared cameras, thermal sensors, radars, frequency sensors and other electronic devices). These capabilities allow to achieve multiple activities, usually in a military environment, as shadowing enemy fleets, target designation and monitoring, location and destruction of land mines, radar system jamming and destruction, and many other applications.

In the civil environment, the UAVs development has become a revolution, because thanks to the technology evolution, the usage of UAVs allows the creation of 3D maps, monitoring building progress, movie production; making every time a widely public interest. In figure A.1, can be observed the expected UAVs growth evolution only in the United States of America, and how to talk about this technology is talk about future.

In many applications described previously, it is needed to guarantee not only efficient, but also secure information delivery. Nevertheless, the existent solutions to provide network safety are limited, like provide only channel layer safety between computers, and not providing safety in a content level.

Bearing in mind this, this present Final Project, approaches this theme, designing and developing a system that is able to provide content level safety, allowing the information transmission through untrusted channels.

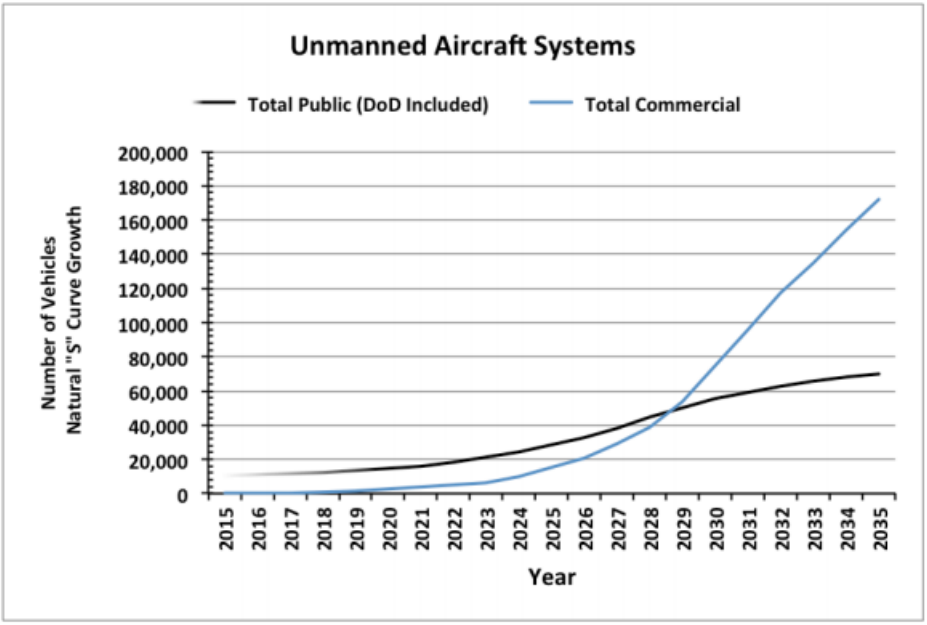

Figura A.1: Predicción del crecimiento de UAVs [11]

A.2. Goals and Project Scope

The purpose of the project is the design and development of a system to deliver secure information in UAVs environment.

- Study the actual safety mechanisms to protect networks communications, so as the new emerging mechanisms of content level safety that are being developed in the new models of ICN. This ICN propose a different from the traditional approach of the Internet, based on accessing the contents regardless of their location. The key idea is the use of names to address content and to forward the content in-network.
- Design and develop an API that allows to deliver information to multiple users in a safe way, verifying the authenticity of the distributed contents.
- Design and develop a set of control mechanisms that allows to request the information to a computer that have it connected to the network. This mechanisms will be based on SIP protocol, standardized by the IETF, to control multimedia communications.
- Validate the development and its viability in a lab environment.

A.3. Document Structure

This document is divided between seven chapters and three appendix.

- Chapter 1, **Introduction**, introduces context, the motivation, and the structure of the project.
- Chapter 2, **State of Art**, introduces the UAVs history, so as new ways of transmissions, introduction to security, multimedia communication protocols, and the regulatory frame of UAVs.
- Chapter 3, **Design**, shows the problem scenario and the requirements to be fulfilled by the API, so as the different modules that take part on the communication.
- Chapter 4, **Implementation**, offers a detailed explanation of the solution proposed in Chapter 3.
- Chapter 5, **Validations and Tests**, lists the different tests executed to check the efficiency and correct performance of the API.
- Chapter 6, **Project Management**, includes the list of task and the time spent, also the economic analysis of the project.
- Chapter 7, **Conclusion and Future Lines**, in this chapter it is commented the results obtained and the issues found during the development of the API. Also, it is introduced future works and improvements of the API.
- Appendix A, **Introduction**, in this chapter it can be found the English translation of the first chapter, as a requirement for the document.
- Appendix B, **Conclusion and Future Lines**, this chapter is composed for the English translation of chapter 7, as a requirement for the document.
- Appendix C, **Extended Summary**, presents an extended summary of the document written in English.

Apéndice B

Conclusion and Future Lines of Work

This chapter includes the general conclusions of the project, such as the different issues when implementing, and the quality of the obtained results; as well how the project could be improved in future line of work.

B.1. General Conclusion

The main goal of this Final Project is the design and development of a system to provide a safe information delivery in UAVs environments. For that reason, the development has been done in a modular way, which includes a *signaling* module, and the API module, which support the secure distribution of content to multiple users.

This project is an actual API, that can be used in any kind of UDP based transmission; for example, video, sensors information or any other telemetric information. It is just needed to reference its methods without any need of previous knowledge about security. Thanks to this API, it is possible to be sure who the sender of the information is, so no one can impersonate the sender, and if someone during the transmission dare to change the information the receiver will know and will reject that data. This API can be used either to transmit information as for the reception of the data.

The implementation of this project has been done by modules due to the different possibilities that this API can provide. The first thing to implement was a simple UDP connection between two clients where the sender could choose the length of the data to send and the delay between packets.

The second step was to introduce security. After many thoughts, it was chosen to encapsulate the generated data after the signature in its own packet, and the use of the algorithm RSA (but in the API is easy to change to another algorithm). With this decision made, it was time to create the pair of keys: the private key, with the purpose of signing; and the public key, in charge of verifying. Both keys are generated by the sender, but the public key is held by the receiver.

The third step was to create a signalling module to inform that a user wants to take part in the conversation. With this purpose, it was chosen to use the signalling protocol SIP. With this process, a user can request to be part of the session, and receive the data.

One of the requirements was to provide an opportunity to re-send the information allowing the final user to verify that the data is actually from the source. For that reason our API allows any user of the communication to act as transmitter, resending the data to all the users that want the information of the source.

After that, as the tests where done in an environment where no errors occur, we had to design a methodology to handle errors and delays in the packets, so when our API is used in a non-perfect environment where errors can occur, it will handle them perfectly fine. This tests were very satisfactory for the author, because creating a hostile environment and to develop the process of error handling was a hard task.

The results obtained from the tests from chapter 5 *Test and Validation*, show that, even with the low performance in the generating a signature process, the performances are suitable for the distribution of telemetric information, even when we generate a signature every packet. As an example, the distributed telemetric information of a system UAV SIVA of INTA, requires a video transmission of a 1 Mbps, while the sent sensors information are in rates of Kbps. In case of a transmission in higher rates, the solution proposed allows to secure the source of the information incrementing the amount of packets in the signature process.

B.2. Future Lines

In this section it is mentioned the future lines that this project can follow. Here are listed some ideas:

- **Run tests in a real UAVs environment.** To this effect, a line work to do would be the text of the developed system using the UAV model available at the Telematics Engineering Department of Universidad Carlos III de Madrid. Due to time limitations, this tests could not been completed.
- **System extension.** This system extension would support the computer equipments discovery who receive the telemetric information, and then can be used to access it in real time. With this system extension, it could be chosen among the applicants to contact.
- **Support of new authentication protocols.** The actual version of the software is bases in the use of RSA and SHA algorithms for the signature generation. In future versions, it could be added new algorithms as for example AES for the sign process, and other types of SHA, as SHA256 for the hash-function.
- **Confidentiality support.** Now, the systems allows to authenticate the source of the received information. A future improvement line would consist to encrypt the source information so, with this approach, it can be guarantee the confidentiality of the information, and develop a mechanism for the distribution of the key to the authorised receivers.

Apéndice C

Extended Summary

C.1. Introduction

Nowadays, the word drone or UAVs is well-known as it is frequently mentioned in the news. A UAV is an aerial vehicle that does not need any in-flight crew, but needs a ground crew to monitor the flight.

There are different types of UAVs depending on the purpose of their design. Some are for operations like search and rescue, monitoring de wildlife and the environment, disaster preventions, and so many others applications. Thanks to these different applications, the predicted growth that is going to experience this kind of technology is represented in figure C.1, noticing a huge increment in the numbers of this vehicles in the years to come.

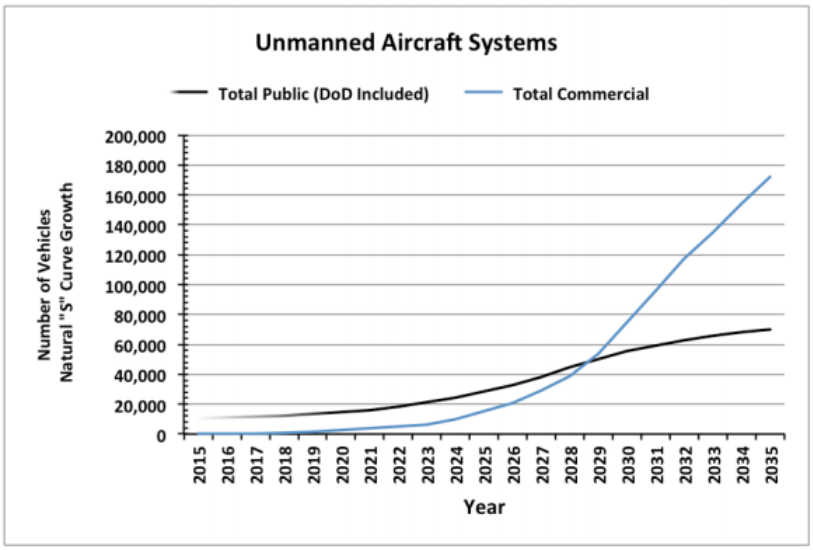

Figura C.1: Prediction of UAVs growth[11]

In many applications, as it was described previously, it is needed to guarantee not only efficient, but also secure information delivery. Nevertheless, the existent solutions so provide network safety are limited, like provide only channel layer safety between computers, and not providing safety in a content level. This Final Project, approaches this goal, designing and developing a system that is able to provide content level safety, allowing the information transmission through untrusted channels.

C.2. System Design

As it has been introduced in the previous section, the main goal is the implementation of a system that allows the distribution of secure information in UAVs environment.

This design will allow an interested user, request and receive telemetric information from a specific UAVs, and authenticate the source of the received information. The system will also allow to request the telemetric information to any other receiver, that will retransmit to the petitioner being able to authenticate the source of the information, the UAV.

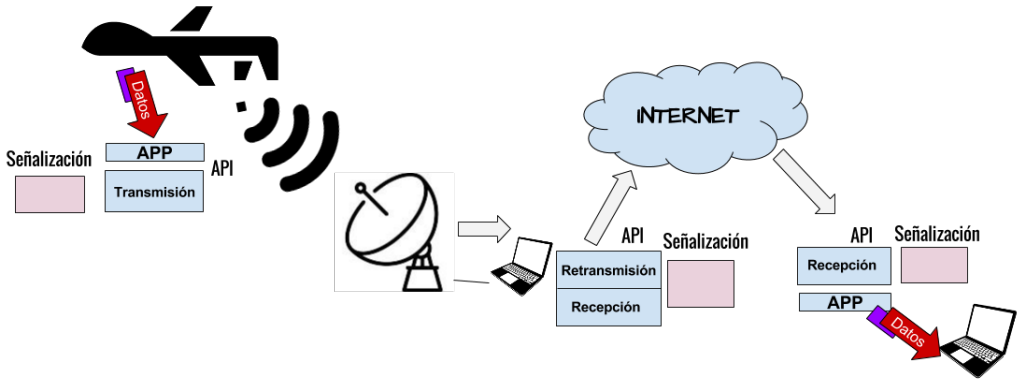

Figura C.2: Problem scenario

The proposed design has two main modules: signalling, and an API. They can be seen in the figure C.2, the signalling is represented with the pink box and the API is represented with the blue box.

Some of the requirements to be fulfilled by the signalling module are: the support of multimedia session establishment with other computers, interact with the API module to configure the retransmission of the information (managing the request connection of the users), based on the SIP protocol.

The API module is divided into two main sub-modules, the transmission and the reception. For the *transmission module* there exists some requirements as to use the UDP protocol in the communication, it has to be able to generate a content level security, when

several users request the information, it has to be able to retransmit the information to all of them. For the *reception module* some of the requirements are: being able to detect if there is some packet missing, if there is no packet missing, it has to be able to verify the signature.

C.3. Implementation

The system development has be done following the design, two main modules, *Signalling* and the API with two sub-modules *Transmission* and *Reception*. The overview of the full implementation can be seen in figure C.3.

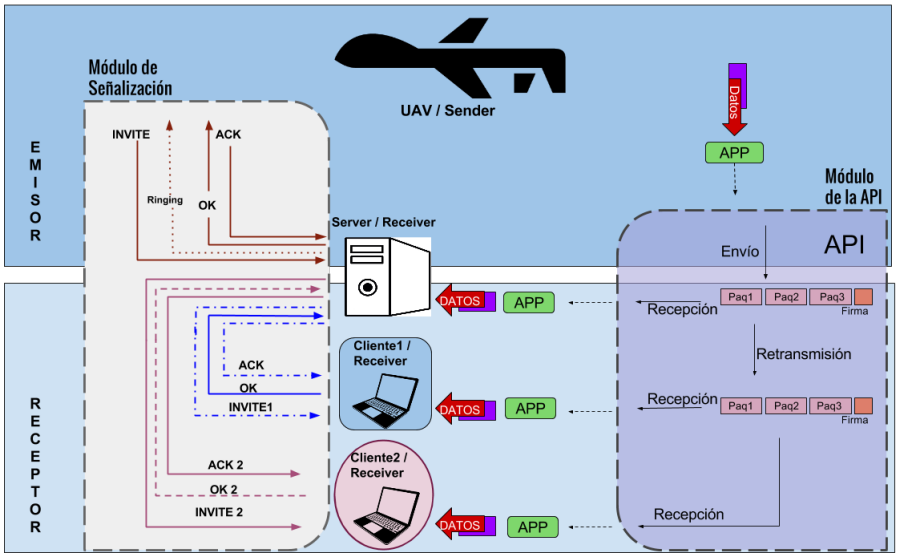

Figura C.3: General Development

The **Signalling module** has been implemented using the JAVA API *Jain Sip version 1.2*. This API provides the necessary classes and methods to implement the SIP protocol in our project. The users applications will generate a SIP Agent using our API **SecureSocket**, that will allow to make use of the SIP protocol. This use will generate some messages, for example, if a user generates a request to receive some data, this user will be sending an *INVITE* message, on the other end, the user that has received the request will accept the request generating an *OK* message. If the user who has sent the request receives the *OK* message, will send back an *ACK* message reporting that a session has been established. This **Signalling module** interacts with the **API module**, managing the users in the communication, and passig to the API module the information about this users. If some user request some data, the *signalling module* will inform the API that it is needed to send to this new incorporation the data from the session.

The **API module** has two sub-modules, one in charge of the transmission of the information using an UDP protocol, and another one in charge of receiving that information.

- The **Transmission Module** will be in charge of transmitting the packets that the *Emitter Application* has passed to him. This module will add a sequence number to enumerate the packets. When a fixed amount of packets have been sent, it will generate a signature with the data from those packets. This signature will be encapsulated in its own packet and will be sent through the UDP protocol. This module is able to duplicate the packets and send them to every user of the communication.

- The **Reception Module** has the mission to receive the packets, and to check if the packets received are in the correct order. If it detects any anomaly, this module is able to take actions to look for the missing packet, and if that packet never comes, it will delete from memory all the received data corresponding to that signature. If everything is correct, it will verify that the data received is actually from the source of the information, making use of the signature verification. This module has to manage also the role of the receivers as the transmitters of the data, passing to the *Trasmission Module* the IP directions of the users, and using that module to send the data.

C.4. Tests

The tests were run in a perfect environment, this means, there was not a chance to have an error in the transmission of the information. Many tests were run in order to check how affects the generation of a signature. The computational force of the computer plays a very important role in this tests, because a computer with low processing capability takes longer to generate a signature.

The way to verify this theory was to check the data rate of the transmission using an application to capture the traffic. The application used for this purpose is *Wireshark*. The different tests that were executed depend on the amount of packages used to generate the signature, from generating a signature every packet to generating a signature every 16 packets. The comparison of the results are shown in the table C.1.

Pruebas	Velocidad	Tiempo Medio	Tasa Efectiva
Prueba I: Firma 1 Paquete	2,47 Mbps	2 ms	2,03 Mbps
Prueba II: Firma 2 Paquetes	4,5 Mbps	1,36 ms	3,71 Mbps
Prueba III: Firma 4 Paquetes	8,77 Mbps	0,8 ms	27,22 Mbps
Prueba IV: Firma 6 Paquetes	12,83 Mbps	0,59 ms	10,56 Mbps
Prueba V: Firma 8 Paquetes	16,83 Mbps	0,45 ms	13,86 Mbps
Prueba VI: Firma 16 Paquetes	32,26 Mbps	0,246 ms	26,57 Mbps

Cuadro C.1: Comparison Table.

Other tests were made in order to check the correct performance of every requirement of the API. Among this tests, it has been made a test to check if the retransmission from a receiver to another user in the communication works, it is shown in figure C.4. In that figure it can be seen that the sender with ip address 202, sends a packet to the receiver 206, and then this receiver 206 sends the packet to another user 207.

23	0.020134	163.117.144.202	163.117.144.206	UDP	173	30000 → 20000 Len=131
24	0.020159	163.117.144.202	163.117.144.206	UDP	1043	30000 → 20000 Len=1001
25	0.020169	163.117.144.206	163.117.144.207	UDP	173	20000 → 50000 Len=131
26	0.020417	163.117.144.206	163.117.144.207	UDP	1043	20000 → 50000 Len=1001
27	0.024007	163.117.144.202	163.117.144.206	UDP	173	30000 → 20000 Len=131
28	0.024032	163.117.144.202	163.117.144.206	UDP	1043	30000 → 20000 Len=1001
29	0.024048	163.117.144.206	163.117.144.207	UDP	173	20000 → 50000 Len=131
30	0.024315	163.117.144.206	163.117.144.207	UDP	1043	20000 → 50000 Len=1001

Figura C.4: Retransmission of a receiver user

Another test that can be displayed with *Wireshark*, is the performance of the **Signalling Module**, shown in figure C.5. This figure represents just the INVITE message, to see more messages from this tests please refer to section 5.5.

```
▼ Session Initiation Protocol (INVITE)
  ▶ Request-Line: INVITE sip:CarlosReceiver@127.0.0.1:5000;transport=udp SIP/2.0
  ▼ Message Header
      Call-ID: 5471f7c0fff7041365d18dc76e3c8419@127.0.0.1
    ▶ CSeq: 0 INVITE
    ▶ From: "carlosSender" <sip:carlosSender@127.0.0.1:3000>;tag=textclientv1.0
    ▶ To: "CarlosReceiver" <sip:CarlosReceiver@127.0.0.1:5000>
    ▶ Via: SIP/2.0/UDP 127.0.0.1:3000;branch=branch1
      Max-Forwards: 70
    ▶ Contact: "CarlosReceiver" <sip:CarlosReceiver@127.0.0.1:3000>
      Content-Length: 0
```

Figura C.5: Invite message

The last tests were made to check if the control of errors in the API worked fine. First, it was necessary no provoke the errors due to that the tests were run in a perfect environment were no errors can occur. As these errors cannot be shown by any application, the figures from section 5.4 show how the errors have been handled.

C.5. Conclusions and Future Lines of Work

This project is an actual API that can be implemented in any kind of UDP based transmission. The API implements a content level security that allows the receiver of the information to verify that the data received is generated by the source of the information. This security is developed with the use of digital signatures. The API will generate a signature of the data sent, sending this signature to the receiver, so this user can verify the data received with the signature received.

It was a design request that the API has to be capable of re-send the information. This approach is useful in the case when a user that does not belong in the conversation requests the information to an user who has the information, making this user a transmitter of the information. The user who has requested the information can verify that the data actually comes from the original source.

After that, as the tests where done in an environment where no errors occur, we had to design a methodology to handle errors and delays in the packets, so when our API is used in a non-perfect environment where errors can occur, it will handle them perfectly fine. This tests were very satisfactory for the author, because creating a hostile environment and to develop the process of error handling was a hard task. Also it was proven with the different tests, that even with the performance of generating a signature every packet is suitable for the distribution of telemetric information.

Some of the future lines of work are:

- **Run tests in a real UAVs environment.** To this effect, a line work to do would be the text of the developed system using the UAV model available at the Telematics Engineering Department of Universidad Carlos III de Madrid. Due to time limitations, this tests could not been completed.
- **System extension.** This system extension would support the computer equipments discovery who receive the telemetric information, and then can be used to access it in real time. With this system extension, it could be chosen among the applicants to contact.
- **Support of new authentication protocols.** The actual version of the software is bases in the use of RSA and SHA algorithms for the signature generation. In future versions, it could be added new algorithms as for example AES for the sign process, and other types of SHA, as SHA256 for the hash-function.
- **Confidentiality support.** Now, the systems allows to authenticate the source of the received information. A future improvement line would consist to encrypt the source information so, with this approach, it can be guarantee the confidentiality of the information, and develop a mechanism for the distribution of the key to the authorised receivers.

Glosario

ACK Mensaje perteneciente al Protocolo SIP que contiene en su Cabecera la configuración negociada para la transmisión.

Best Effort Designa un tipo de servicio en el que la red no puede garantizar que los datos lleguen al destino, ni ofrecer una calidad de servicio determinada.

Cabecera Parte inicial de un mensaje o paquete, que normalmente contiene información para el control y encaminamiento del mismo.

Cifrado Transcripción en letras o símbolos, de acuerdo con una clave, un mensaje o texto cuyo contenido se quiere proteger. [16].

Cifrar Consiste en la alteración de la señal original, de tal forma que no pueda ser reconocida, empleando un código secreto.

Criptoanálisis Método que compromete la seguridad de un criptosistema.

Criptografía Arte de escribir con clave secreta o de un modo enigmático. [16].

Criptosistema Agrupamiento de diferentes conjuntos, de posibles textos planos, posibles textos cifrados, con un espacio de claves y para cada clave existe una regla de cifrado y una correspondiente regla de descifrado.

Data Rate Tasa de datos, número de bits de información transmitidos por segundo..

Datagram Packet Encapsulación de datos necesaria para la transmisión utilizando protocolo UDP.

Dirección Secuencia de elementos binarios que indican el destino final de una comunicación o de un conjunto de datos.

Downlink Enlace de bajada, enlace de comunicación para la transmisión de señales de radio desde una plataforma en movimiento a una estación terrestre.

Firma Mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la entidad originadora de dicho mensaje.

Gigabit Corresponde a 10^9 bits.

Internet Sistema global de interconexión de redes de computadores que hace uso de Protocolos para conectar millones de dispositivos mundialmente.

INVITE Mensaje predefinido del protocolo de conversación SIP, utilizado para empezar una conversación.

IP El protocolo estándar Transmission Control Protocol (TCP)/IP que define el datagrama IP como la unidad de información a través de internet y que provee la base de para la conexión sin previo acuerdo, entrega de paquete con servicio Best Effort.

JAVA Lenguaje de programación característico por "write once, run everywhere".

Multicast Grupo de comunicación donde la información es direccionada a un grupo de equipos simultáneamente.

OK Mensaje predefinido del protocolo de conversación SIP, utilizado para negociar en el extremo del receptor.

OSI Modelo de interconexión de sistemas abiertos.

Padding Relleno.

Protocolo Conjunto de normas y reglas utilizado para establecer y mantener una comunicación de datos entre las diversas estaciones que componen un enlace.

Puerto Abstracción usada por los protocolos de transporte TCP/IP para distinguir entre múltiples destinos dentro de un equipo.

RINGING Mensaje predefinido del protocolo de conversación SIP, utilizado para informar al emisor que el INVITE ha llegado al receptor.

Router Equipo dedicado a una función especial el cual une dos o más redes y envía paquetes de una a otra.

Seed Clave usada para inicializar un generador de números pseudoaleatorios para generar otras claves..

Signature Mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la entidad originadora de dicho mensaje.

Socket Abstracción provista por el sistema operativo UNIX que permite a una aplicación acceder a los protocolos TCP/IP.

TCP Protocolo a nivel de transporte TCP/IP que provee de un servicio de transmisión de confianza, full dúplex del cual dependen muchos protocolos del nivel aplicación. Es un protocolo orientado a conexión..

Texto plano Datos que pueden leerse y entenderse sin necesidad de usar ninguna herramienta complementaria. [2].

Throughtput Tasa a la que un ordenador o red envía o recibe datos..

UAV Unmanned Areal Vehicle.

UAV SIVA del INTA El Sistema Integrado de Vigilancia Aérea (SIVA), es un sistema para uso civil y militar, desarrollado por el Instituto Nacional de Técnica Aeroespacial (INTA), compuesto por un UAV y una GCS [3].

UDP User Datagram Protocol, protocolo que permite a una aplicación en una máquina enviar un datagrama a otra aplicación en una máquina remota. Utiliza IP como protocolo de red, es un protocolo de transporte y no orientado a conexión..

Unicast Transmisión de mensajes a una única red destino identificada por una dirección única.

Uplink Enlace de subida, enlace de comunicación para la transmisión de señales de radio desde una estación terrestre a una plataforma en movimiento.

URI Cadena corta de caracteres que identifica inequívocamente un recurso.

Bibliografía

[1] Expertos británicos descifran el código secreto del ejército alemán. [Online]. Available: http://pe.tuhistory.com/hoy-en-la-historia/expertos-britanicos-descifran-el-codigo-secreto-del-ejercito-aleman

[2] *An Introduction to Cryptography.* Network Associates, Inc, 2002.

[3] (2011) Siva. [Online]. Available: http://web.archive.org/web/20110915135501/http://www.inta.es/doc/programasaltatecnologia/avionesnotripulados/siva.pdf

[4] (28 August 1980) Rfc:768 user datagram protocol. [Online]. Available: https://tools.ietf.org/html/rfc768

[5] (Abril 2012) Cripto-análisis sobre métodos clásicos de cifrado. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=4289408

[6] (July 2003) Rfc:3550 rtp: A transport protocol for real-time applications. [Online]. Available: https://tools.ietf.org/html/rfc3550

[7] (July 2006) Rfc:4566 sdp: Session description protocol. [Online]. Available: https://tools.ietf.org/html/rfc4566

[8] (June 2002) Rfc:3261 sip: Session initiation protocol. [Online]. Available: https://www.ietf.org/rfc/rfc3261.txt

[9] (September 1981) Rfc:791 ip: Internet protocol, darpa internet program, protocol specification. [Online]. Available: https://tools.ietf.org/html/rfc791

[10] (September 2001) Rfc:3174 us secure hash algorithm 1 (sha1). [Online]. Available: http://www.ietf.org/rfc/rfc3174.txt

[11] (September 2013) Unmanned aircraft system (uas), service demand 2015-2035. [Online]. Available: https://fas.org/irp/program/collect/service.pdf

[12] Atul Kahate, *Cryptography and Network Security.* McGraw-Hill Education, 2013.

[13] G. Camarillo, *SIP Demystified.* McGraw-Hill, 2002. [Online]. Available: http://www.voiceip.com.ua/lit/McGraw-Hill%20-%202002%20-%20SIP%20Demystified.pdf

[14] G. de España, Ed., *Real Decreto-ley 8/2014, de 4 de julio, de aprobación de medidas urgentes para el crecimiento, la competitividad y la eficiencia.*

[15] Douglas E. Comer, *Internetworking with TCP/IP.* Prentice Hall, 1995.

[16] R. A. Española. (2016) `http://www.rae.es/`.

[17] Gabriel M. Brito, Pedro Braconnot Velloso, Igor M. Moraes, *Information-Centric Networks, A New Paradigm for the Internet.* Wiley, 2013.

[18] J. Jonsson, B. Kaliski. (February 2003) Rfc:3447 public-key cryptography standards (pkcs) 1: Rsa cryptography specifications version 2.1. [Online]. Available: https://tools.ietf.org/html/rfc3447#page-27

[19] Manuel J. Lucena López. (27 mayo de 2011) Criptografía y seguridad en computadores. [Online]. Available: http://wwwdi.ujaen.es/mlucena/lcripto.html

[20] Oracle. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html

[21] E. Proulx, *An Introduction to the JAIN SIP API*, 10/17/2007. [Online]. Available: http://www.oracle.com/technetwork/articles/entarch/introduction-jain-sip-090386.html

[22] Reg Austin, *UNMANNED AIRCRAFT SYSTEMS. UAVS Design, Development and Deployment.* Wiley, 2010.

[23] Thaer El Gamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* IEEE Transactions on information theory, 1985.